Keith Mayes
Konstantinos Markantonakis *Editors*

Smart Cards, Tokens, Security and Applications

Second Edition

Springer

Smart Cards, Tokens, Security and Applications

Keith Mayes · Konstantinos Markantonakis
Editors

Smart Cards, Tokens, Security and Applications

Second Edition

Springer

Editors
Keith Mayes
Director of the Information Security Group, Head of the School of Mathematics and Information Security
Royal Holloway, University of London
Egham, Surrey
UK

Konstantinos Markantonakis
Smart Card Centre, Information Security Group
Royal Holloway, University of London
Egham, Surrey
UK

ISBN 978-3-319-50498-8 ISBN 978-3-319-50500-8 (eBook)
DOI 10.1007/978-3-319-50500-8

Library of Congress Control Number: 2016959517

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

I would like to dedicate this book to Susan, George and Amelia, to my mother and to the memory of my late father

Keith Mayes

I would like to dedicate this book to Maria, Eleni and Georgios

Konstantinos Markantonakis

Founders Message

The ISG Smart Card Centre (SCC) was established in 2002 as a centre of excellence to complement the world leading work of the Information Security Group (ISG), which was established in 1990. With the ISG having pioneered information/ cybersecurity for more than a quarter of a century, the SCC still feels quite new, but at 14+ years old it is clearly something of enduring substance. This is testament to the vision and commitment of the SCC Founders, who saw the need for a UK-based centre involved in academic research, training and expert advisory activities, in the areas of smart cards, RFIDs and embedded systems security. Of course technology is always changing, and today, the SCC is also interested in the security of Near-Field Communication (NFC) mobile phones/devices, vehicular/transport security and the enormous challenges of safeguarding the Internet of Things (IoT), and critical infrastructure in general. However, the need to consider implementation security and attack resistance, as well as secure design, is more vital than ever. This is a principle that we have instilled in the many M.Sc. students who have studied and completed projects in the SCC, with the aid of the course text book. The book was originally published in 2008 and came about because no other text could offer the breadth and depth of content needed for the M.Sc. Today, the book is a reference found on many bookshelves (physical or virtual!) beyond academia and this new edition aims to keep the content fresh, relevant and useful.

The Founders continue to be proud of their association with this book and of their pioneering efforts that brought about the SCC.

The Founders:

Professor Michael Walker OBE
Founder Director of Vodafone Group Research and Development
Professor of Telecommunications (Vodafone Chair)
Royal Holloway, University of London

Dr. Klaus Vedder
Group Senior Vice President
Giesecke & Devrient GmbH

Professor Fred Piper
Founder Director of the Information Security Group
Royal Holloway, University of London

Professor Keith Mayes
Founder Director of the ISG Smart Card Centre
Director of the Information Security Group
Head of the School of Mathematics and Information Security
Royal Holloway, University of London

Foreword

The idea of inserting a chip into a plastic card is nearly as old as public-key cryptography. The first patents are now 40 years old, but practical, massive application deployment started only twenty years ago due to limitations in storage and processing capacities of past circuit technology. Today, new silicon geometries and cryptographic processing refinements lead the industry to new generations of cards and more ambitious applications.

Over the last two decades, there has been an increasing demand for smart cards from national administrations and large companies such as telephone operators, banks and insurance corporations. More recently, other identity and payment markets have opened up with the increasing popularity of home networking and Internet and the advent of International Civil Aviation Organization (ICAO) passport standards.

At the same time, cryptographic hardware engineering grew into an active research field with flagship conferences and journals such as the Conference on Cryptographic Hardware and Embedded Systems (CHES) and the Journal of Cryptographic Engineering. Over a hundred Ph.D. theses on side-channel attacks are defended each year throughout the world and attack contests regularly benchmark the community's level of knowledge.

The traditional carrier for a conventional smart card is a plastic rectangle on which can be printed information concerning the application or the Issuer (even advertising) as well as readable information about the cardholder (as for instance, a validity date or a photograph). This carrier can also include a magnetic stripe or a bar code label. An array of eight contacts is located on the micromodule in accordance with the ISO 7816 standard, but only six of these contacts are normally connected to the chip, which is (usually) not visible. The contacts are assigned to power supplies, ground, clock, reset and a serial data communication link (commonly called I/O). However, over the last years, mainstream protocols and technologies such as USB, http and Simple Object Access Protocol (SOAP) were adopted by the card industry and the cards form factor has evolved. In the same time, contactless cards have become increasingly popular. New Trusted Execution

Environments have recently matured and start to be deployed in smartphones. These can be regarded as non-detachable smart cards that, in essence, are subject to the same design and security constraints as those of usual smart cards.

Current smart card CPUs range from simple 8-bit microcontrollers to sophisticated 32-bit architectures. RAM capacities, historically limited to a few hundreds of bytes, are steadily increasing. ROM and Electrically Erasable Programmable Read-Only Memory (EEPROM) are being progressively replaced by Flash, whilst native execution is commonly substituted by Java applets.

Cutting one's way through such a technology-rich environment requires several years of industrial experience or a very thorough reference, such as this book.

Throughout the years of research in Royal Holloway's Smart Card Centre, Kostas and Keith have invented, implemented and benchmarked an incredible number of card technologies. Their industrial background and academic approach are, to my knowledge, unique in the field. The number of their alumni is impressive.

I hope that you will enjoy reading and learning from this book as much as I did.

August 2016

Prof. David Naccache
Laboratoire d'informatique de l'Ecole normale supérieure
Université Paris II, Panthéon-Assas

Preface

When the first edition of this book was published back in 2008, the scope was anything to do with smart cards and security tokens in the widest sense. The aim was in fact to provide a complete story, looking at a cross section of technologies, processes, applications and real-world usage. The original motivation for the book was to provide a suitable reference text for the Masters course in Information Security, run by the Information Security Group (ISG) at Royal Holloway, University of London (RHUL). However, as the planning for the book advanced we realised that various industries and government departments can become quite narrow in their understanding of smart cards/RFIDs and that looking across industry and across roles (such as technical, business and logistics) could be beneficial for a much wide range of readers. Eight years on, in this second edition, we find we have new material to cover, whilst surprisingly, very little of the old material is redundant. Although smart cards and RFIDs are still at the core of what we do, increasingly we are involved in general embedded systems, mobile device security, trusted execution and the all-encompassing Internet of Things (IoT). Indeed, the course for which the book was written is now called "Smart Cards, RFIDs and Embedded Systems Security" and our Smart Card Centre is now the "Smart Card and IoT Security Centre"!

One constant across the years is that to deliver such breadth of information requires input from many experts and so we are very pleased and proud of the calibre of the authors and reviewers that have made this book possible. We hope that you will enjoy this book and find it a useful guide and reference.

Structure of the Book

This book consists of eighteen chapters. Each chapter is a completely autonomous contribution in a chained discussion which aims to bring researchers, practitioners and students up to speed with the recent developments within the smart card arena. In order to enhance the reader experience, each book chapter contains its own abstract, introduction, main body and conclusion sections. Furthermore, bibliography resources can be found at the very end of each chapter. The following list provides a more detailed overview of the topics that are discussed in the different chapters of this book.

Chapter 1 provides an introduction to a very wide range of smart card-related issues. It surveys the different types of cards, tokens, and it also considers the main types and capabilities of popular applications utilising smart card technology. The chapter is considered as a good starting point for newcomers to the field and perhaps those that have perhaps focussed on one business or technical area.

Chapter 2 discusses the different steps in the smart card production chain. The analysis covers all the main steps during the smart card manufacturing phase starting with the production of the card body, chip moulding and smart card personalisation and delivery. Finally, it concludes with current and future trends and challenges.

Chapter 3 provides an overview of the most widely utilised smart card operating systems and platforms that enable multiple applications to be securely managed and reside in the same smart card.

Chapter 4 discusses the role of the Subscriber Identity Module (SIM) in the mobile telecommunications industry and describes the associated standards. It presents the authentication and ciphering processes in some depth and provides a practical comparison between the two technologies prior to exploring further value-added service and toolkit features. Finally, it provides some insight into the future evolution of technology.

Chapter 5 examines the role of smart card technology within the financial payments industry. It examines how the credit card industry has evolved over the decades and explains some of the issues with magnetic stripe card technology. Subsequently, it presents the main features of smart card technology in the light of the EMV card specifications. The discussion continues with 3D-secure and token authentication.

Chapter 6 deals with the issues around content protection in the satellite TV industry. In particular, it examines the commercial motivation as the driving force behind content protection, how smart card security is utilised in order to provide the necessary functionality and finally highlights how a typical pay-TV system operates.

Chapter 7 provides an overview of the Trusted Platform Module (TPM) and highlights commonalities and differences with smart cards. It provides an introduction to the security mechanisms provided by the TPM and provides a guide to the associated standards and literature.

Chapter 8 explains how Common Criteria evolved, how it is defined and how it is used in practice. More importantly, it examines how Common Criteria is applied to the complex and demanding field of smart card security evaluations.

Chapter 9 focuses on the various attacks and countermeasures that apply to smart cards. As many applications rely on cryptographic algorithms for sensitive operations, this chapter focuses on the attacks that could affect smart cards performing cryptographic operations. Furthermore, it provides references to the corresponding countermeasures and emphasises the need for rigorous design, implementation and test of cryptographic algorithms and their underlying host platforms.

Chapter 10 provides a brief overview of the wide range of issues associated with the smart card application development processes. In particular, it examines the development of an application for the popular Java Card platform. It also highlights practical issues around application development and monitoring tools. Finally, it looks into development of the mobile phone applications that can exploit SIM and USIM card capabilities by using it as a trusted security element.

Chapter 11 analyses the use of the smart card within the telecommunications industry as a managed platform. It examines how the mobile phone operators are using the necessary tools and technology in order to remotely update and enhance, over-the-air, the functionality of SIM and USIM cards.

Chapter 12 provides a valuable introduction to the main standards used to manage and access smart card readers connected to personal computers. Their main functionality is analysed and attached code samples aim to provide a detailed overview, but also to enable the reader to reuse them in order to quickly develop sample host applications that will communicate with smart cards. Mobile phone Application Program Interface (API) that can access smart cards, SIM card and Secure Elements is also included in this chapter.

Chapter 13 provides an introduction to the Radio Frequency Identification (RFID) concepts and also summarises the aspects most relevant to contactless smart card systems. Several different systems along with the operating principles are described. The chapter also provides an overview of the main Radio Frequency (RF) interface and communication theory along with the various RF standards. The chapter concludes with an overview of Near-Field Communication (NFC).

Chapter 14 explains how national requirements for eID cards and e-passports can be realised by utilising physical, logical and hardware functionality. Furthermore, it highlights the importance and requirements of the relevant standards.

Chapter 15 first examines the historical use of technology in smart cards before highlighting the future trends. It looks into the different options and choices which can be made within a smart card scheme along with the issues which affect the design of the card and its applications. Finally, it discusses issues around consumer demand and the drivers that will define the smart card technology of the future.

Chapter 16 describes how security challenges arise from a rapidly growing population of smart and/or embedded digital devices that can leverage, monitor and control systems and components in the physical world—known as the Internet of Things (IoT). This chapter outlines the means for addressing these challenges and introduces a proposed overall decision-making framework for developing an IoT security strategy.

Chapter 17 discusses the concepts behind the MULTOS smart card operating system and introduces the development environment and tools. It provides several examples of code to assist in explanations. It also briefly outlines possible future uses for MULTOS outside of smart cards.

Chapter 18 explains how a Trusted Execution Environment (TEE) provides an execution platform on a mobile device, isolated from the rest of the operating system and other applications: the chapter explores what constitutes a TEE and their various security features. Standardisation efforts related to TEEs and example implementations of TEEs are also outlined. Host Card Emulation (HCE) provides an alternative to “traditional” Near-Field Communication (NFC) card emulation by allowing an application on the host CPU of a mobile device to emulate a card and communicate directly with an external reader. HCE introduces new security risks to the mobile ecosystem, and this chapter illustrates how these can be managed to an acceptable level.

In order to make reading of this more convenient, we also provide a subject index at the very end of the book.

June 2016

Keith Mayes
Konstantinos Markantonakis
Royal Holloway, University of London, Egham, UK

Acknowledgements

We offer our sincere thanks to all those that helped bring about the second edition of this book. There are too many to mention them all, but a few names must appear here. Firstly, we would like to thank Mike Walker, Klaus Vedder and Fred Piper, for their vision and steadfast support of the Smart Card Centre (SCC). We must certainly not forget the companies that have sponsored the SCC over the years and especially Vodafone, Giesecke & Devrient, Transport for London, the UK Cards Association, ITSO and Orange Labs UK; as without them, there would be no SCC and no book. A book also needs authors and reviewers, and ours all deserve medals for their excellent contributions and their tested patience. Last, but definitely not least, we must marvel at the herculean efforts of Sheila Cobourne and Danushka Jayasinghe in helping to bring this book to print.

Contents

Editors and Contributors

About the Editors

Prof. Keith Mayes, B.Sc., Ph.D. CEng FIET A. Inst. ISP is the Director of the Information Security Group (ISG), and Head of the School of Mathematics and Information Security at Royal Holloway, University of London, which has been pioneering information/cybersecurity research and education since 1990. He is an active researcher/author with 100+ publications in numerous conferences, books and journals. His current research interests are diverse, including mobile communications, Near-Field Communication (NFC), mobile platform security, smart cards, Radio Frequency IDs (RFIDS), the Internet of Things, transport ticketing/system security, embedded systems and e-commerce. Keith joined the ISG in 2002, originally as the founder Director of the ISG Smart Card Centre, following a career in industry working for Pye TVT, Honeywell Aerospace and Defence, Racal Research and Vodafone. Keith is a Chartered Engineer, a Fellow of the Institution of Engineering and Technology, a Founder Associate Member of the Institute of Information Security Professionals, a Member of the Licensing Executives Society and an experienced company director and consultant.

Prof. Konstantinos Markantonakis B.Sc., M.Sc., MBA, Ph.D. (London) received his B.Sc. in Computer Science from Lancaster University in 1995, his M.Sc. in Information Security in 1996, his Ph.D. in 2000 and his MBA in International Management in 2005 from Royal Holloway, University of London. He is currently a Professor of Information Security in the Information Security Group in Royal Holloway, University of London. He is also the Director of the Information Security Group Smart Card Centre (SCC). His main research interests include smart card security and applications, secure cryptographic protocol design, key management, embedded system security and trusted execution environments, mobile phone operating systems/platform security, NFC/RFID/HCE security, grouping proofs, electronic voting protocols. He has published more than 140 papers in international conferences and journals. Since completing his Ph.D., he has worked as an independent consultant in a number of information security and smart card related projects. He has worked as multiapplication smart card manager in VISA International EU and as a Senior Information Security Consultant for Steer Davies Gleave. He is a member of the IFIP Working Group 8.8 on Smart Cards. Since June 2014, he is the vice-chair of IFIP WG 11.2 Pervasive Systems Security. He continues to act as a consultant on a variety of topics including smart card security, key management, information security protocols, mobile devices, smart card migration program planning/project management for financial institutions, transport operators and technology integrators.

Contributors

Raja Naeem Akram is working as a postdoctoral Research Assistant at the Information Security Group (ISG), Royal Holloway, University of London, and is involved with the research projects involving digital avionics and payment systems. Previously, Raja worked as research fellow at the Cyber Security Lab, Department of Computer Science, University of Waikato, New Zealand. At the Cyber Security Lab, he was involved with the user-centric security and privacy paradigms. Before joining the University of Waikato, he worked as a Senior Research Fellow at Edinburgh Napier University. During his work at the Edinburgh Napier University, he worked on the RatTrap project. The RatTrap project was involved in designing a suite of preventive technologies to avoid online fraud—especially in the online affiliate marketing. Raja obtained his Ph.D. in Information Security from Royal Holloway, University of London. He completed his M.Sc. Information Security at Royal Holloway, University of London in September 2007. He also has an M.Sc. Computer Science from University of Agriculture, Faisalabad and B.Sc. (Mathematics, Physics and Geography) from University of the Punjab, Lahore. His research interests revolve around the user-centric applied security and privacy architectures, especially in the field of smart cards, and data provenance in a heterogeneous computing environment. In addition, he is also interested in smart card security, secure cryptographic protocol design and implementation, smartphone security, trusted platform architecture and trusted/secure execution environment.

Tony Boswell began working in IT security as a security evaluator in one of the original UK government Evaluation Facilities in 1987. Since then, he has worked on a wide range of secure system developments and evaluations (including the ITSEC E6 certifications of the Mondex purse and the MULTOS smart card operating system) in the government and commercial domains. Tony has been involved in UK and international interpretation of evaluation requirements for smart cards since 1995 and continues to contribute to multinational technical community work on interpretation and maintenance of Common Criteria evaluation requirements, as well as assisting developers to take their products through Common Criteria evaluations. He is currently a senior principal consultant at DNV GL and technical manager of the DNV GL Technical Assurance Laboratory CLEF.

Joos Cadonau received his B.Sc. in Electronic Engineering and Telecommunication in 1994 from the University of Applied Sciences in Bern, Switzerland. After his degree, he worked as project manager for the primary Swiss Telecommunication provider Swisscom AG in Network Access Management Systems, followed by a project manager role in Ascom AG in the area of PBX switches (Private Branch Exchange). From 2001 until 2011, he acted as Product Manager for Sicap AG, a Swisscom subsidiary, managing SIM-based over-the-air solutions for telecommunication customers all over the globe. During this period, he focused on the role of the SIM card in complementary security centric services, such as NFC payments, M2M connectivity management platforms and Mobile Identity products. After further engagements for the Swiss Railway company in the area of GSM-R and for Swisscom AG in the Centre of Competence for Machine-to-Machine communication, he joined iQuest Schweiz AG in 2014 to develop Identity and Internet of Things (IoT) offerings.

Sheila Cobourne, MA (Oxon), MBA., M.Sc. read Physics at Lady Margaret Hall, Oxford, and went on to work as a systems analyst developing financial systems for various multinational organisations. She has an MBA from Henley the Management College and a M.Sc. in Mathematics from the Open University. She studied the M.Sc. in Information Security at Royal Holloway, University of London and passed (with distinction) in 2010. She is a Ph.D. student in the Information Security Group Smart Card Centre supervised by Professor Keith Mayes.

Paul Dorey, Ph.D., CISM F.Inst.ISP is a cybersecurity and risk management strategist, who supports company Boards, Executives and CISOs in devising and developing their cybersecurity management capability. He has over 25 years of management experience in digital security and enterprise risk management including information security, digital security of process control/SCADA systems, Business Continuity Planning, privacy and information management. His leadership roles include Global CISO at BP and global leadership of strategy, information security and risk management functions at Morgan Grenfell and Barclays Bank. Paul has consulted to several governments and was a founder of the Jericho Forum and for several years sat on the Permanent Stakeholders Group of the European Network Information Security Agency (ENISA). He was one of the founders of the Institute of Information Security Professionals, and after 5 years as Chairman of the Board, he is now a Fellow of the Institute and Chairman Emeritus. He is also a Visiting Professor at Royal Holloway, University of London.

Claus Ebner born 1962 in Krumbach, Germany, studied mechanical engineering at the Technical University of Munich. Afterwards, he did his doctorate at the Institute for Machine Tools and Industrial Management of the same university. In 1995, he joined Giesecke & Devrient (G&D) in Munich as Head of IT and development in the Card Service Center. After his involvement in the introduction of an ERP system for the card business, he took over responsibility for the international production software development of G&D. This centre provides software for smart card production to the worldwide subsidiaries of G&D and supports the setup of card production or personalisation sites in G&D's solution business as well. He then moved to the Corporate IT of G&D as Head of Enterprise Architecture, working on the standardisation of the global IT landscape. Now, as Head of IT Governance & Architecture at G&D, he is also responsible for managing the business demands and the IT project portfolio.

Tim Evans is an international industry-recognised SIM expert who has delivered many SIM designs and patents for companies including Vodafone and Truphone. Tim is very involved in SIM/UICC/USIM/eSIM standardisation, where he was chair of ETSI SCP REQ and vice-chair of ETSI SCP over 10 years. Tim has trained many operator SIM teams and is currently working for Vodafone R&D. He is Vodafone's 3GPP SA3 and ETSI SCP delegate setting the security standards for 5G, Internet of Things (IoT) and future SIM technologies; the rapporteur of around 30 SIM and security standards and also is the Vodafone eSIM solutions architect.

Thomas Goetz was born 1970 in Neumarkt i. d. OPf., Germany, and studied microengineering at the Georg-Simon-Ohm University of Applied Sciences of Nuremberg. Afterwards, he did his thesis at the R&D Center of Philips Kommunikations Industrie (PKI) in Nuremberg. In 1994, he joined Giesecke & Devrient in Munich as project manager in the card service centre. From 1999 to 2001, he was based at the company's London office for the set-up and integration of a service centre serving the UK market. Then, he took over responsibility for the Munich-based international smart card production support and industrial engineering team. Since 2011, he responds the strategic planning and its deployment for Global Operations of the Mobile Security Business Unit. The unit supplies innovative security solutions for digital applications to financial institutions, network operators, transport providers and businesses in all sectors.

Gerhard P. Hancke is an Assistant Professor at City University of Hong Kong. Previously, he worked as postdoctoral researcher and fellow in the Information Security Group, Royal Holloway, University of London. He obtained M.Eng. and B.Eng. degrees from the University of Pretoria (South Africa) in 2002 and 2003, and a Ph.D. in computer science with the Security Group at the University of Cambridge's Computer Laboratory in 2008. His research interests are system security, embedded platforms and distributed sensing applications.

Danushka Jayasinghe obtained his B.Sc. (Hons) in Computer Networking from the University of Greenwich. He completed his M.Sc. in Information Security (with Distinction) at the Information Security Group, Royal Holloway, University of London, in 2013. He joined the Smart Card Centre to follow a Ph.D. under the supervision of Professor Konstantinos Markantonakis. His research interests are on modern electronic payment systems with the consideration of security, privacy and efficiency of the underlying payment protocols and platforms. He is also interested in alternative payment schemes other than conventional card-based payment methods including digital cash and anonymous and fair-exchange payment protocols.

Ingo Liersch entered the smart security industry in 2001 when joining Giesecke & Devrient GmbH (G&D). As a Product Manager he focused on Government electronic identity cards including smart card operating systems and applications. In

2004, his role changed to a project manager. He was in charge of eID and eHealth Card projects in Europe and Asia. From 2007 until 2013, he was responsible for segment marketing in the G&D Division Government providing solutions for secure government documents. His work included presales, business consultancy, market intelligence and marketing communication for the business segments identity and health care. Furthermore, he was responsible for emerging products such as encrypted mobile phones and brand protection systems. Ingo joined Infineon in the Business Line Government Identification in 2014. He is responsible for Product Marketing for the Infineon microcontroller platforms for electronic documents such as ePassports, eIDs or eHealth cards.

Ingo has many years of experience in secure identification and authentication, and together with his team he has become a trusted adviser to governments. Ingo holds a degree in Electrical Engineering and in Industrial Engineering. He is active in the Smart ID industry and participates in various industry organisations. Ingo represents Infineon in Silicon Trust (www.silicon-trust.com). Ingo gives lectures on electronic passports and ID cards at Royal Holloway, University of London (Information Security Group Smart Card Centre). Furthermore, he is well known in the smart security market due to publications and articles in identity journals.

David Main began his career as an electronic engineer in the late 1970s and progressed with a number of companies from guided weapons, through process control, to passport and fingerprint systems. Highlights included working with the first 16-bit microprocessors and a contactless electronic coin. In the early 1990s, he moved to Visa and became involved with chip technology from the inception of EMV and developed personally in the areas of cryptology and information security. EMV focus topics included: chip to terminal security architecture, Level 1 communications, contactless (including NFC/mobile) and supporting standardisation ISO & European. David now runs a small independent consultancy specialising in payment industry technology and security. His interests include travel, gardening, golf and in particular vintage car restoration.

Damien Sauveron received his M.Sc. and Ph.D. degrees in Computer Science, from the University Bordeaux 1, France. He is Associate Professor with Habilitation at the XLIM laboratory (UMR 7252 University of Limoges/CNRS France) since 2006. He is Head of the Computer Science Department of Faculty of Science and Technology of University of Limoges. Since 2011, he is a member of the CNU 27, the National Council of Universities (for France). Since 2014, he is chair of IFIP WG 11.2 Pervasive Systems Security and was vice-chair before.

His research interests are related to Smart Card applications and security (at hardware and software level), RFID/NFC applications and security, mobile networks applications and security (especially Unmanned Aerial Vehicle (UAV)), sensors network applications and security, Internet of Things (IoT) security, Cyberphysical Systems security and Security Certification Processes. In December 2013, the General Assembly of IFIP (International Federation for Information

Processing) has granted him the “IFIP Silver Core Award” for his work. He has been involved in more than 100 research events in different positions (PC chairs, General Chair, Publicity Chair, Editor/Guest Editor, Steering Committee Member, Program Committee Member, etc). At the time of writing, he is involved in more than 5 funded projects on security of UAV fleets.

Chris Shire has a background in security technologies and semiconductor hardware. He joined Infineon (then Siemens) Chipcard & Security business line in 1998, with many years of experience in the industry. His current focus of activity is with projects in the embedded, mobile, transport and payment sectors. He is active in helping to set standards for the UK and establish new security solutions. Chris is an active member of the Institution of Engineering and Technology (IET), UK Smart Card Club, and has been a guest lecturer for several years on the Royal Holloway, University of London M.Sc. course for Smart Card Security. He has written several articles on security technology and contributed to textbooks on the subject.

John Tierney gained his degrees in Pure Mathematics and Computing from Sheffield (B.Sc.) and Numerical Analysis from Liverpool (Ph.D.) Universities. Dr. John Tierney initially worked in communications security, including early ITSEC work in the early 1990s. He has worked with smart cards since 1993, initially developing a secure operating system and applications for prepaid phone card. He joined Mondex in 1999 and worked on a series of Common Criteria and ITSEC evaluations on a range of smart card products. Since 2002, he was worked for MasterCard providing support to banks who implement chip, contactless and mobile payment technologies. He lives in Wirral.

Allan Tomlinson received a B.Sc. in Applied Physics from the University of Strathclyde in 1981; M.Sc. in Microelectronics in 1987 and doctorate in 1991, both from the University of Edinburgh. His thesis was on “VLSI architectures for cryptography”. He then joined the Institute of Microelectronics at the National University of Singapore, working on secure NICAM broadcasting and video compression. In 1994, he moved to General Instrument in California to work on the Digicipher II Conditional Access system for digital video broadcasting. Before joining the Information Security Group at Royal Holloway, he was Principal Engineer at Barco Communications Systems where he was responsible for the development of the “Krypton” DVB Video Scrambler. He also served for a number of years on the DVB Simulcrypt committee. His current research interests are distributed systems security, trusted computing and mobile network security.

Chris Torr joined the MULTOS Consortium, MAOSCO Ltd, as technical manager in 2012 bringing with him eleven years’ experience in smart cards. His career has also spanned the electronics and defence industries. It has taken him around the world promoting, developing and deploying solutions and providing training.

He has a B.Eng. Honours degree in Information Systems Engineering from Coventry University.

Michael Tunstall has been working in embedded security since 1998, primarily focused on side channel and fault analysis of cryptographic devices. He started his career in Gemplus Card International, the world's leading smart card maker, and was involved in the development of side-channel attacks and countermeasures, from the moment it was introduced to the cryptographic community as a very real security issue. He is also responsible for some of the first publications demonstrating that fault analysis of cryptographic devices is possible and that suitable countermeasures are required in all such devices. He has coedited a book entitled "Fault Analysis in Cryptography", which is a summary of the state-of-the-art in fault analysis and is the first text book on this topic in the literature.

Assad Umar received his B.Sc. in Business Information Technology from Coventry University UK in 2010 and his M.Sc. in information security from Royal Holloway in 2012 with a distinction. Prior to joining the Information Security Group as an M.Sc. student, he worked for a year in the Information Security Department of the Nigerian Communications Commission NCC. His research interests include smart cards, Near-Field Communication, smartphone security, transport ticketing, network security and security architectures. He is a Ph.D. student in the Smart Card Centre conducting research in transport ticketing systems under the supervision of Professor Keith Mayes on a project funded by Transport for London.

Gary Waite is the executive director—embedded SIM, Connected Living at GSMA. In the early 1990s, Gary designed and sold what became the de facto standard SIM and mobile test tool for the entire GSM industry via a start-up attracting $20m of venture capital funding to enhance SIM card security. Gary spent over ten years with Telefonica, advising on key technology areas such as SIM, location and Machine-to-Machine (M2M). His recent work has made the GSMA Embedded SIM specification the de facto standard for the M2M industry, and today, he is playing a key role in the development of remote SIM provisioning for consumer devices.

Anjia Yang is a Postdoctoral Fellow in the Department of Computer Science at City University of Hong Kong, where he obtained his Ph.D. in 2015. He visited I2R, A*STAR, Singapore (2013–2014), and Chalmers University of Technology, Sweden (2014), respectively. His research interests include RFID security and privacy, Security of Internet of Things, Cloud Security and Cryptographic Protocols.

List of Figures

List of Tables

List of Reviewers

Keith Mayes
Konstantinos Markantonakis
Raja Naeem Akram
Tony Boswell
Sheila Cobourne
Paul Dorey
Thomas Goetz
Gerhard Hancke
Danushka Jayasinghe
Ingo Liersch
David Main
Damien Sauveron
Chris Shire
Allan Tomlinson
Chris Torr
Michael Tunstall
Assad Umar
Anjia Yang

Chapter 1
An Introduction to Smart Cards

Keith Mayes

Abstract When we released the original version of this book, back in 2008, we stated that the concept of a smart card was not particularly new, but that the practical use of smart cards in a range of diverse applications had never been more popular. Eight years on and the statement is still valid, although we have seen some trends towards certain types of smart card and applications, and indeed more focus on embedded smart/secure chips that do not rely on the card form factor. Furthermore, we have seen the introduction of Near Field Communication (NFC), which permits mobile phones to emulate smart cards and readers. This chapter provides a first introduction to a wide range of smart cards and tokens, considering the various types, capabilities, popular applications and the practicality of their development and deployment, covered in detail within subsequent chapters.

Keywords Smart cards · Tokens · Security · Applications · Java · MULTOS · RFID · SIM · ID Contactless · Microprocessor cards · Chip card · Magnetic Stripe card · Memory card · Development · Lifecycle · Tags · IoT · NFC · MIFARE Classic

1.1 Introduction

Smart cards, and in particular the specialist chips within them, are perhaps some of the most widely used, but underestimated electronic security devices in use today. In many cases these devices are in the front line, defending citizens and systems alike against attacks on information[1] security. Because they have tended to be small and often concealed, smart cards have carried on their important work, largely unnoticed, but this is changing. High profile use of smart cards for IDs [1], passports [2], credit cards [3] and e-tickets [4] means that the smart card is now a regular topic for the

[1]Note within this book we use the terms information security and cyber security interchangeably.

K. Mayes (✉)
Director of the Information Security Group, Head of the School of Mathematics and Information Security, Royal Holloway, University of London,
Egham, Surrey TW20 0EX, UK
e-mail: keith.mayes@rhul.ac.uk

K. Mayes and K. Markantonakis (eds.), *Smart Cards, Tokens, Security and Applications*, DOI 10.1007/978-3-319-50500-8_1

popular press. With all this activity and positive momentum, one would expect that the term smart card has a clear definition and the physical devices would be easy to identify. Unfortunately this is not the case and ambiguity abounds, so the first priority in this book is to provide some clarity and definitions to be used within the chapters.

1.2 What Is a Smart Card?

It is perhaps a little surprising to start a smart card text book with such a trivial sounding question as "What is a smart card?". However, judging by some press articles and even technical reports, the answer seems to elude even the great and the good.

Normally at this point, a text book would dive into the ancient history of card evolution, which basically says we have such wonders because some rich guy forgot his wallet one day. However, this can add to the confusion regarding what is, or is not, a modern-day smart card, so the history trip will come later once we have a few definitions to work with.

Part of the problem stems from the use of *smart*. If a system is much more convenient because a particular card is being used then that is a pretty smart thing to do, even if by technical standards the card is unremarkable. The next problem comes from *card* which to most people would imply say a credit card sized piece of plastic, whereas various sizes are possible and indeed the innards of the device could be embedded in something completely different, such as a passport or phone.

The candidates that could be described as smart cards are therefore numerous and so our definition will be refined a little to weed out some of the least relevant.

A smart card

1. has a unique identifier,
2. can participate in an automated electronic transaction,
3. is used primarily to add security and
4. is not easily forged or copied.

   In support of (2) and (3), two more definitions will be added, i.e.
5. can store data securely,
6. can host/run a range of security algorithms and functions.

This definition will now be applied to a few well-known card types to see if they are truly *smart*.

1.2.1 Magnetic Stripe Cards

Magnetic stripe cards are still widely used in a range of applications. They are characterised by being low cost and relatively easy to read/write. An example is

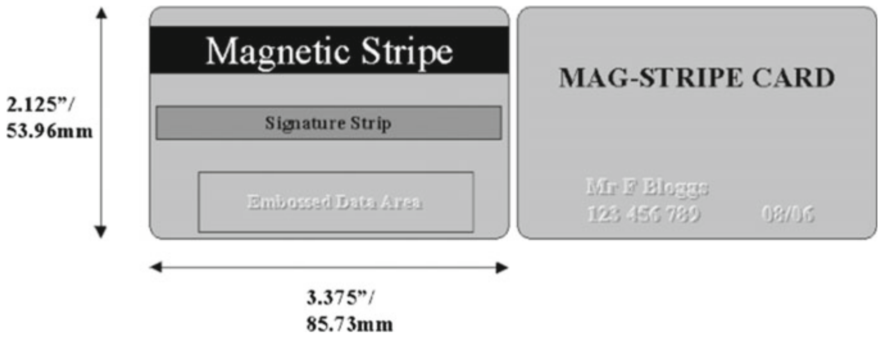


Fig. 1.1 A typical magnetic stripe card

shown in Fig. 1.1. For many years this type of card was used for credit and debit card financial applications, although in Europe it is being phased out by EMV cards described in Chap. 5. The cards are used throughout the world and indeed for a diverse range of applications including entitlement cards, tickets and access control systems.

In terms of our smart card definitions, the magnetic stripe card can be regarded as follows:

- It has a unique identifier,
- it is clearly involved in electronic transactions and
- in many cases it is *meant* to provide some security element; unfortunately it is very poor when tested against the fourth definition, as it can be copied or forged.

Considering Fig. 1.1 in closer detail, we have a piece of plastic which is used as a carrier for a stripe of magnetic tape. The plastic card may also carry some text or images designed more for human interpretation and checking rather than the electronic transaction that is of primary interest. Invisible to the human eye is the information stored within the magnetic stripe. The stripe is not dissimilar to that used in an old cassette recorder, i.e. a strong magnetic field controls the alignment of magnetic dipoles into various orientations along the length of the tape. The alignment is preserved even when the polarising field is removed and so information is stored by the dipoles. The alignment can be simply tested and the information recovered by a tape reader head. Because the tape and the equipment used are relatively crude, the information storage capacity is quite limited. To maximise this in a practical manner, multiple tracks are stored along the stripe, again similar to an audio tape recording. On each track one can store a few bits of identity-related information and the method of storage is known as Wiegend [5] format. A fairly exhaustive description of these cards is beyond the scope of this book and the curious reader can consult another reference [6].

The important thing to say about magnetic stripe cards is that they are not smart cards because they fail the fourth smart card definition and quite disastrously too. It is therefore astonishing to see how long they survived in financial applications. The root of the problem is relatively easy to find. The magnetic stripe is not much more than a piece of audio tape and so it can be easily read and indeed rewritten with relatively simple equipment. This means that it is quite trivial to forge/clone magnetic stripe cards. A lot of effort has gone into making the plastic carrier harder to duplicate (although with limited success), but there is not much that can be done about the magnetic stripe used in the automated transactions. Two types of magnetic stripe fraud [7] have become legendary:

- Skimming—here the information from a valid card's magnetic stripe is copied to another card for use in fraudulent automated transactions.
- Counterfeiting—here the plastic carrier/card is very carefully copied, but the magnetic stripe may be blank or invalid.

How a skimmed card may be exploited is fairly obvious, but counterfeiting deserves a few word of explanation. Although there are various countermeasures on cards to discourage counterfeiting, such as special graphics, embossed printing and holograms, they really just represent more inconvenience and time for an attacker, rather than serious obstacles. The counterfeit is to fool a human operator rather than an automated process, so how can this be useful when most physical transactions tend to be automated? The reason is that it is a very common occurrence for the magnetic stripe on a valid card to be unreadable due to wear and tear. This has lead to an acceptance for the manual fall back mechanism. For example, a person goes into a petrol station to buy fuel. The assistant swipes the card once, twice, rubs the stripe on his/her sleeve and tries once more but in vain. He/she then simply reads the numbers printed on the card plastic and types them into the point of sale terminal to complete the transaction. For Internet purchases it is even easier as there is no attempt at an automated process and the attacker only needs to have read the required information from the source card, rather than create the counterfeit.

Because of the prevalence of skimming and counterfeiting, magnetic stripe cards are no longer be recommended to safeguard significant financial transactions. Far better is the electronic chip-based solution standardised by EMV and described in detail within Chap. 5. The EMV solution is now the standard in some countries such as the United Kingdom where it has drastically reduced fraud for card-holder-present transactions [8] (e.g. physical transactions in a store). However, EMV has not yet delivered similar safeguards for the increasing volume of Internet transactions which is now the most worrying form of fraud and hence the industry focus for countermeasure development.

There can be a future for magnetic stripe cards where the solutions are very cost sensitive and the cards are not protecting anything of significant value, for example a loyalty card or a library access card. The lack of reliability of the stripe is perhaps not a problem when the cards are only required to have a limited lifetime and usage. For all other applications the trend is towards the use of electronic chips embedded within the card, in order to improve, functionality, reliability and security.

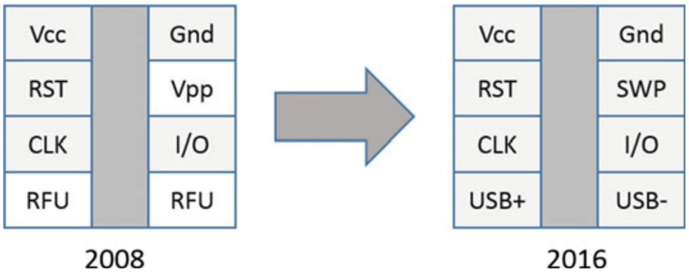


Fig. 1.2 Contacts of a chip card

1.2.2 Chip Cards

As the name suggests a chip card is basically a plastic card, rather like the magnetic stripe card, that has an electronic chip embedded in it. Historically, these cards were easy to identify by virtue of the contacts that were usually gold or silver coloured. The use of these contacts is described within standards [9] and a traditional example is shown on the left-hand side of Fig. 1.2. As a minimum, the contacts provide power to the chip via Vcc/GND, a communications I/O line, plus clock (CLK) and reset (RST) lines. The Vpp pin was once used for re-programming old-style EEPROM memories, but is now obsolete. Because the two lower pins were historically unused and Reserved for Future Use (RFU), some cards and readers only make use of the six upper contacts. However, some industries have already standardised the use of the old RFU and Vpp contacts for new services. Referring to the right-hand side of Fig. 1.2, we can see how mobile communication standards have re-specified the spare pins for a high-speed USB interface and for a Near Field Communication Single-Wire Protocol (SWP) [10], as described in Chap. 4.

A contact chip card is accessed by placing it within a card reader which simply makes physical contact with the metal pads, allowing the chip to be powered, clocked and for communication to take place. The low-level electrical interface is described in ISO standards [9]. Communication is half-duplex, with the reader/host taking the role of master and the card the slave. There are in fact two low-level protocols that can be used to exchange information, known as T = 0 and T = 1 and both are defined in ISO standard [9]. The T = 0 protocol is favoured by the mobile communication industry and is simply a byte interface where information is sent a byte at a time, rather like communicating with a modem over a serial link. By contrast, T = 1 is a block protocol favoured by financial institutions.

One could fall into the trap of assuming that all chip cards are more secure than magnetic stripe cards and that the presence of the gold contacts implies that this is a smart card. These assumptions are incorrect and dangerous.

The simplest chip card could contain a single fixed value. The application protocol would simply be to read this fixed value for comparison. It would be trivial for an attacker to read the value from a valid card and produce a copy and so this type of card fails our fourth smart card definition in the same way as the magnetic stripe card.

Another type of card is the memory card that may be used to perhaps keep a count of purchased telephone call minutes or accumulated loyalty points. Such cards might not have added security and so contents may be easily read and copied. Moreover, the memory may be rewritten to undermine the application or change IDs. The simple memory card also fails our fourth requirement for a smart card as it can be easily copied or modified. Note that there are memory cards that incorporate fixed functionality security (often proprietary), which we may refer to as secured memory cards (e.g. MIFARE DESFire EV1) used in popular systems. However, as we are interested in more general-purpose security devices the focus moves towards microprocessor chip cards. Sometimes these chips are referred to as secure microcontrollers rather than microprocessors; we will use the terms interchangeably within this book.

1.2.3 Microprocessor Chip Cards

Microprocessor cards have the scope to satisfy the requirements for a smart card because they are not only able to store and communicate stored values, but can do so within the context of security protocol programmes. The protocol interface to the device can be defined such that it is logically impossible to extract information, or to reprogramme the contents without appropriate permissions which are checked and enforced by cryptographic functionality. At first glance one would think that the search is over and that the presence of a microprocessor chip card is synonymous with a smart card, but this is not the case. A conventional non-specialised microprocessor card can give you logical security, but that is insufficient for a smart card. Attackers are not put off by logical security, but would employ a range of techniques that attack the chip directly or exploit information leakage from an operating device. These attacks are described in detail within Chap. 9, but suffice to say that a microprocessor used in a smart card is designed in a very specialised way to be "tamper-resistant" [11]. This is perhaps the most important property to remember about a smart card. When considering conventional smart cards our definition could be as shown below.

- A smart card contains a tamper-resistant microprocessor chip (incorporating countermeasures against known attacks) that is difficult to forge or copy. It includes a unique ID, can participate in automated electronic transactions, can store data securely and run/host a range of security protocols and algorithms.

The consideration so far has focused on traditional smart cards that make use of electrical contacts to the chip. However, there is growing interest and usage for cards that do not have physical contacts, but exploit radio techniques instead.

1.2.4 Contactless Smart Cards and RFIDs

The smart card industry tends to talk about cards with contacts or contactless smart cards; however, industries that have been concerned with tagging, product coding and tracking, tend to talk about Radio Frequency Identification (RFIDs). If the definition of a smart card has been much abused then so too has RFID. Because of this someone may refer to a contactless smart card as an RFID or vice versa. This section will try and remove some of the ambiguity, but there will still be a grey area of overlap.

In principle, RFID is very simple to define. It is a device that presents an ID to a reader device via Radio Frequency (RF) means. It does not imply any protocol security, clone prevention, or tamper resistance; however, some real-world devices may incorporate such measures. It might even be argued that a magnetic stripe card should be regarded as a RFID, as it makes use of an electromagnetic field to communicate, although its probably best to ignore this possibility as there is enough confusion already.

A chip card that presents an ID (perhaps for door access) could therefore be considered as a special case of an RFID.

Generally, RFIDs tend to be less sophisticated than contactless smart cards (although there is no fundamental technical reason for this) and are not restricted to the physical card format. Whilst lower layer protocols tend to be standardised [12], a worrying aspect is that the application layer is often proprietary and the design information is held secret. From a security perspective the reliance on secret proprietary protocols and algorithms is always treated with much scepticism and nowadays we have a *classic* example to show why.

1.2.5 The MIFARE Classic

When we were writing the first edition of this book we knew a few things about the MIFARE Classic [13] product from NXP.

- It used a proprietary secret algorithm (CRYPTO1).
- It was intended as a medium (rather than high) security memory card.
- It was very widely used.
- It was quite old, and newer products were available.
- Its keys were small (48-bit) compared to best practice.
- There were data sheets (from 2004) of seemingly *unauthorised* products.

With such small keys the security was reliant on the secrecy of the CRYPTO1 design, which violates Auguste Kerckhoffs principle [14] and is a example of the much criticised *security by obscurity*. If the algorithm becomes known then it can be coded into a key cracker to determine the secret keys, and therefore the existence of the data sheets was worrying. It suggested that no later than 2004, the product design, including the algorithm, had already been reverse engineered or otherwise copied. In simple terms, the demise of this product was a question of when and not if. It actually came about from some more reverse engineering [15], but this time the goal was publication rather than commercial gain. Publishing the algorithm was enough to compromise the product; however, once exposed to expert scrutiny, major flaws were discovered, as is often the case with ageing proprietary designs. To cut a long story short, a key cracker was not necessary; a simple laptop would do. It was possible to tease secret keys out of readers and eventually access and modify keys and data in legitimately issued cards. In defence of the MIFARE Classic it was a very popular and useful product, and when first launched there was little to better it as a contactless secured memory card. It was geared towards speed and usability rather than high security, and NXP had higher security alternatives at the time the Classic was hacked. The system owners that used the products bear some responsibility as although the algorithm was secret, the key size was known to be 8-bits smaller than the already obsolete single-key DES, so would not have passed a system security review.

1.2.6 Smart Tokens

In this book we may mention smart tokens. A smart token can be considered as a personalised device that has all the useful security, functional and tamper-resistant properties of the smart card, but is not provided in a normal plastic card format. Interestingly the smart card that is most prevalent in the world, the mobile Subscriber Identity Module (SIM), might be regarded as a smart token. Whilst it started life in the full-card format and is still produced by a card manufacturing process, it is commonly used in the plug-in format, see Fig. 1.3, although mini and nano formats are now in use that are even smaller than the plug-in, being not much bigger than the contact module. Just about any format is possible and a range of communication and powering methods could be considered.

One thing that is clear from looking at the types of device and their names is that there is a lot of overlap. The literal explanation of the device name and or acronyms might once have been simple and accurate; however, the names have

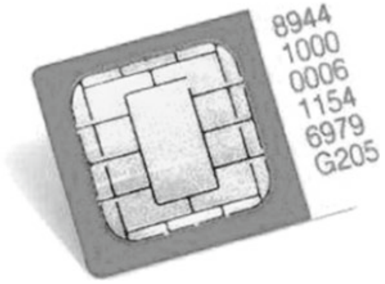


Fig. 1.3 A plug-in format SIM card

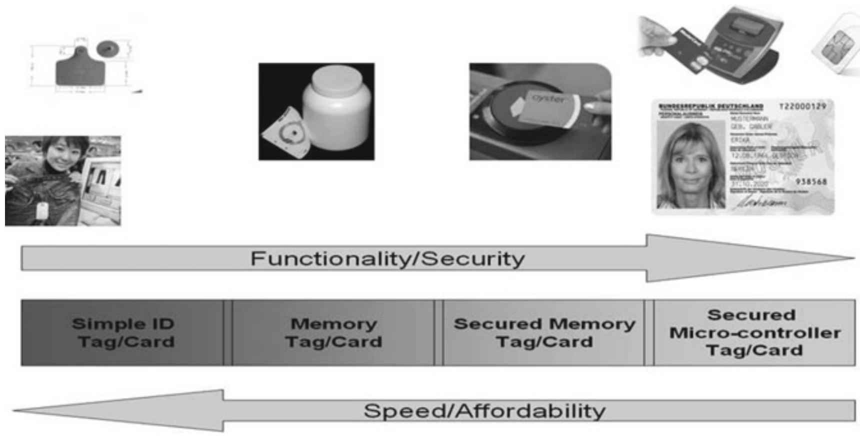


Fig. 1.4 Smart card-RFID range and trade-offs

evolved in the public perception, e.g. no one thinks of a card that is smart in some way, but rather of a smart card. Similarly it is doubtful that anyone will remember that RFID is strictly an ID presented via radio frequency means, because RFID has simply become a modern word. Most often we are concerned with passive contactless cards/RFIDs that extract their power from the readers' electromagnetic field and then modulate the field to communicate. A range of these devices are illustrated (along with contact card types) in Fig. 1.4. On the left-hand side of the figure we see the simpler types that may just present an ID without and security protection, as might be found in simple product or animal tagging. The memory cards might have very simple access control or none at all, put permit storage of information additional to an ID such as description and place and date of issue. The contents of memory cards can be protected by cryptographic Message Authentication Codes (MACs) and kept confidential by means of cryptography; however, a major weakness is that the contents may be copied, swapped or used in clones. Secured memory tags may offer mutual authentication and strong confidentially, integrity and access protection; however, a careful choice of product is required to avoid the kind of problems that arose with the MIFARE Classic vulnerabilities. The right-hand side of Fig. 1.4 illustrates the most significant types of secured microcontroller devices such as those found in passports, bank cards and mobile phones. One might imagine that using the more inferior types is just due to cost constraints; however, this is not always the case. For example, smart transport tickets have very tight operational time constraints in order to avoid throughput and safety issues at station gates, and it is easier to achieve this with a fast secured memory card than a sophisticated microcontroller. If you are fortunate to have a service or system where security risk is not an issue then the ID cards will do; giving you not only a cheap/fast tag, but avoiding key management and reader security issues. You may also be constrained in your choice by other factors such as temperature, waterproofing, lifetime, robustness and tolerance to read range and shielding.

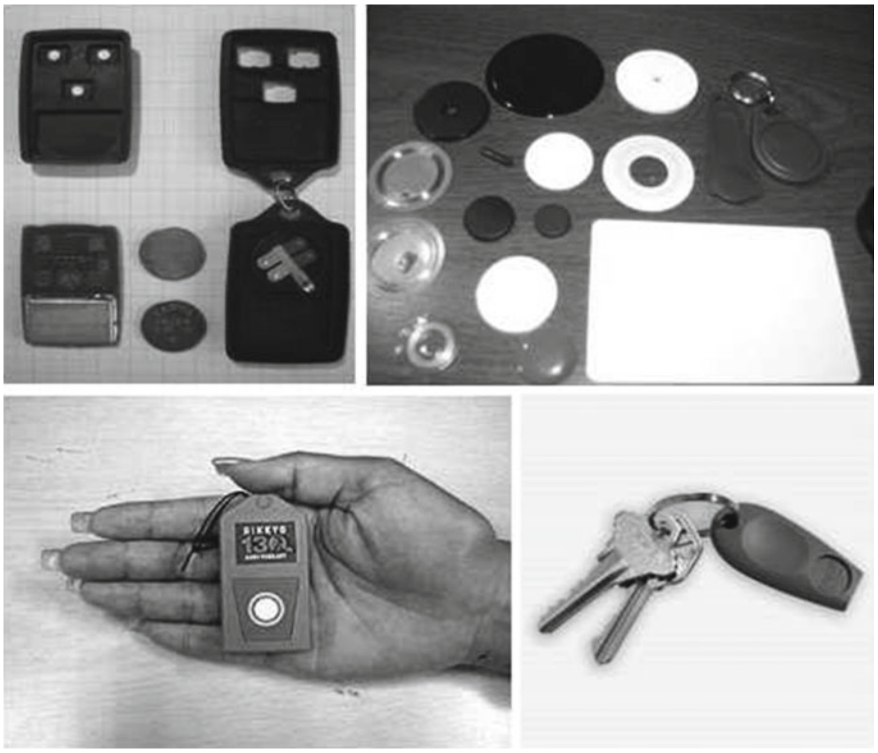


Fig. 1.5 Active RFIDs

There are also active RFIDs that incorporate their own power sources. A common example is the keyfob for remote central locking of a car. Active RFIDs/tags also tend to be used where passive devices would struggle, for example, situations requiring a long read range or where there is a lot of metal that can adversely affect the electromagnetic fields used by passive devices. Some conventional examples are shown in Fig. 1.5. Although it is not normally thought of this way, the mobile phone is the most widespread active RFID, as it has a unique ID, a power source and its own radio transmitter. Furthermore, the mobile phone is a very sophisticated and location aware RFID, with multiple radio bearers, support for cryptographic protocols, application hosting and a direct user interface. This has led to some privacy concerns, because for a mobile network to function it has to dynamically track the location of mobile phones (and hence their users) in order to route calls.

Just in case we do not have enough names and confusion, we must also be aware of Near Field Communication (NFC) [10]. The simplest way to describe this is as a contactless/RFID interface for a mobile phone. Essentially this now standardised functionality allows the phone to act as a contactless smart card/RFID or indeed as the reader to communicate with an external card. For the card emulation aspects a hardware Secure Element (SE) is normally used, which has much in common with a conventional attack-resistant smart card chip, but can be provided in the phone in a variety of ways; within the SIM itself, as a separate chip on the phone Printed Circuit Board (PCB), or as a chip in a plug-in memory card. It is also possible to have a software emulation of a SE, such as in Android Host Card Emulation (HCE), although the attack-resistant capabilities of this approach are considered inferior to specialist hardware.

For the remainder of this book, unless there is a particular need to differentiate devices, the name smart card may be used to represent all types of device incorporating a tamper-resistant microprocessor chip supporting secure data storage and security functions/algorithms. In most cases the actual physical format and low-level communications interfaces will be ignored. This serves to illustrate that what is of real interest and value is the chip at the heart of the device. The term RFID may be used for simpler radio frequency tags.

1.3 Smart Card Chips

The microprocessor found in a smart card bears little resemblance to the processor you will find in a modern PC although the core of the smart card chip is not dissimilar to some of the PC's early ancestors. Smart card chips tend to be very small and so there is always a limit to what functionality and resources can be crammed in. There are several historical reasons for the size limitation. First, the cost of a chip is proportional to the area of silicon used and although individually, this may not be a lot, smart cards may be ordered in their millions and so are always very cost sensitive. Second, because smart cards are often delivered via the normal post they must withstand bending and twisting stresses. If the chip is too large then these stresses will break the chip or the wires connecting the chip to contact pads or antennas. Third, as the chip gets larger and more complex, its power requirement would normally increase. Host devices tend to be quite miserly when supplying power and even if they were not, there could be heat dissipation problems within the module. Finally, because of its security purpose, the essential code and functionality should be kept as small and as simple as possible, so it can be exhaustively reviewed and tested to the stringent levels required by standardised evaluation criteria. The amount of code that any CPU must trust in order to enable a secured operation is known as the Trusted Computer Base (TCB), and in a smart card this is very small indeed.

When the first edition of this book was written we were wondering whether in the future, some of the restrictions on smart cards might ease, especially for mobile communication smart cards, with the capability for devices with large memories, faster interfaces and processors [16]. While some of this has appeared in standards, practical adoption is harder to find. In fact the smart card has been kept steadfastly simple, with manufacturing and technology advances being used to mainly reduce the size and cost of chips. Therefore, we will stick to conventional chips and an old-style layout is shown in Fig. 1.6. As mentioned previously, the device has no clock or in-built power supply, but it does have a CPU and three types of memory as well

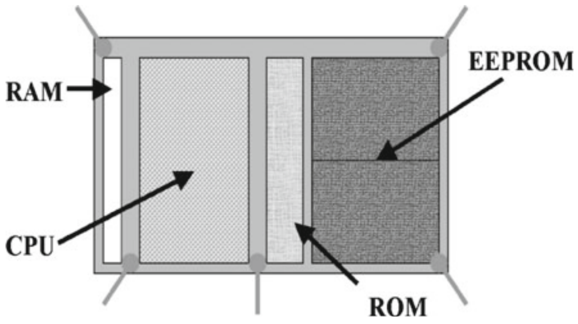


Fig. 1.6 A smart card chip (old)

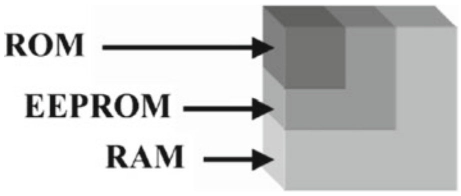


Fig. 1.7 Comparison of chip area needed for various memory types

as the interconnecting circuitry. Understanding the types of memory is important as what is stored where is usually a trade-off decision for the designer.

The Read Only Memory (ROM) is some times referred to as the chip mask. It is characterised by the fact that its contents (which are identical for all cards in the batch) can only be set/written to, during card production and then only read during normal card operation.

The Random Access Memory (RAM) is used for fast dynamic storage of programme run-time variables and the stack. In this respect it is similar to the RAM in a PC although there is much less of it. When power is removed the RAM loses its contents.

Electrically Erasable Programmable Read Only Memory (EEPROM) is very useful as it can be programmed after manufacture and does not lose its contents when the power is removed, i.e. it is non-volatile.

Looking at Fig. 1.6 one could easily get the wrong idea about the likely proportions of RAM, ROM and EEPROM. This arises because the memories have different packing densities, i.e. how many bits fit into a given silicon area. ROM packs best of all so is an area and cost efficient method of storage; however, the fact that it cannot be rewritten is a serious drawback, making it useless for user data and flexible function storage. For well-tested common functionality and constant data it is fine. Next best for efficiency is the EEPROM, which can be used for all kinds of data and application storage. RAM packs poorly and is very restricted, which is a challenge for programme developers that is only likely to get worse as application sophistication increases.

A typical relationship between the memory types and areas is shown in Fig. 1.7.

We must also mention another important type of storage known as flash memory. This is well known for memory sticks, but it took a while to be adopted in smart cards.

Flash memory takes the place of the ROM and the EEPROM and is becoming the dominant non-volatile memory type in smart cards. It has a number of advantages that make it attractive when compared to EEPROM. First, an EEPROM memory ages, i.e. it can only be written to a certain number of times (few hundred thousands), which can limit the lifetime of certain card applications. A flash memory does not have this problem and is also much faster to write. The designer does not have the same ROM/EEPROM dilemma as the split can be adjusted during the design phase, although it must lock down the flash equivalent to ROM space after production, for security reasons. RAM is still at a premium, as for the traditional smart card case. For the manufacturer there are also advantages as it is not necessary to create many different customer masks as the different configurations can be soft loaded. One potential drawback is that any soft loading may take extra time in production and whereas the chip cost is proportional to chip area, the cost of the final smart card is proportional to the production time. Finally, some card Issuers still have concerns about the security of flash memory technology and may simply forbid its use in their product specifications; although this viewpoint is becoming less common.

For now we will stick with the concept of real RAM, ROM and EEPROM within our smart card device, although remembering that flash can replace both ROM and EEPROM. The use of RAM is quite evident and so the first question to consider is what can be put in the ROM for maximum benefit. Typically, the ROM would contain the operating system, well-tested common functions and constant data. The EEPROM can be used for user data, user programmes as well as general application data and functions that require modification during operational life. Sometimes a proportion of the EEPROM is used for functionality that would just not fit in the ROM and/or patches to compensate for the bugs and extensions of the ROM programmes. The EEPROM is used for essential non-volatile application data storage, including personalisation information.

EEPROM is a very useful resource, especially as there are techniques to update its contents whilst in the field. Therefore, to create a future proof device one would try and issue a larger smart card with plenty of spare EEPROM to accommodate fixes and new functionality. However, this increases the chip area and hence cost, which means it will be resisted by whoever is signing the cheque for the next million devices. The battle between the alien mindsets of designers/strategists and purchasers is quite a common occurrence. On the one hand you have the purchasers arguing for savings today, whereas the designers/strategists push to spend more now to avoid future problems from built in obsolescence. Unfortunately modern business has a very short-term focus and so the head-in-the-sand savings argument wins far too often.

1.4 Tamper Resistance

One of the key strengths of the smart card (which really means the chip) is its tamper resistance, i.e. the ability to resist known and anticipated attacks. Looking at the

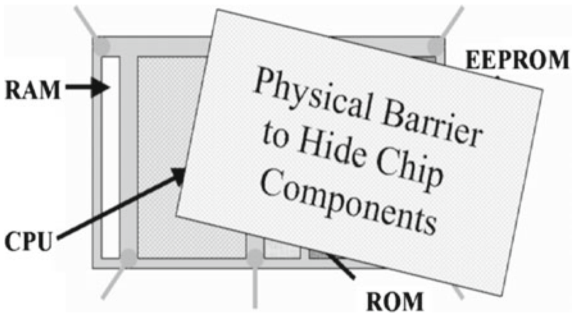


Fig. 1.8 Smart card chip anti-probing layer

picture in Fig. 1.6 one might think that the attackers job is not so hard as its very easy to identify the attack targets such as memories and busses that might yield valuable secrets and permit card cloning. However, this is not the case and the chip layout shown is very old and presented for clarity of explanation. In reality the first thing that an attacker may encounter is a physical layer of silicon that prevents probing of the circuit (see Fig. 1.8).

If this can be overcome then there may be an active current-carrying layer, so that any break renders the chip useless to the attacker. Get beyond this layer and you may find that the circuitry has been scrambled making it difficult to find the attack target. If a bus or memory is eventually found then it may well be encrypted. Clearly if an attacker has expensive equipment, expertise and is prepared to destroy many cards, he may eventually extract some information, but unless this is a global secret arising from a bad design it is difficult for much advantage to be gained. Smart card attacks and countermeasures are covered in detail within Chap. 9 but for now it suffices to say that the smart card chip mounts a very robust defence, even when faced with sophisticated attacks from well-equipped experts.

1.5 Smart Card Characteristics

So far, a fairly glowing report has been given of smart cards; however, like any device they have both strengths and weaknesses. In order to exploit smart cards appropriately it is just as important to appreciate the weaknesses as well as the strengths. Table 1.1 presents this is a summary form. Note that the memory capacities and CPU capabilities are just typical indications, as they evolve over time.

Bearing in mind that the smart chip might only be less than 9 mm^2 then the processor and memory capabilities are surprisingly good. However, compared to a PC or mobile phone processor the card is rather feeble and would not be a good choice for handling large amounts of data or time critical processing. However, cards with co-processors are no slouches when it comes to specialised cryptographic processing. The main positive features include the tamper-resistant security, the precise standardisation and the resulting consistency and control. Obvious weaknesses include

Table 1.1 Summary of smart card strengths and weaknesses

Features	Limitations
CPU (16–32 bit)	**Helpless Alone**
RAM (4–16 kb)	No internal power supply
ROM/Flash (128–256 kb)	Externally restrictions on power consumption
EEPROM (64–256 kb)	No user interface
Crypto-processor option	No clock
Very small	**Limited (by PC comparison)**
Low power	Memory
Low cost	CPU speeds
Secure	**Issued device**
Standardised	Legacy cards may be inflexible
Operating systems	New cards require deployment
Development tools	
Multiple suppliers	
Consistent and Controllable	

the fact that the card is helpless on its own and thus is always reliant on other system elements. For example, a conventional smart card has no internal power source, no direct user interface and (with few exceptions) not even a clock. From a system management perspective another significant feature is the ability to personalise the smart card to a particular customer or account. Smart cards tend to be issued in very large numbers and so one of the ever present problems is dealing with legacy devices. A great new service that only works on newly issued cards may take years to reach a large proportion of the customer base. Legacy problems can be minimised by forward-looking design and the use of lifecycle management systems; however, legacy problems are often “designed-in” to satisfy short-term cost savings.

One of the features that could be described as an advantage or limitation, depending on your viewpoint, is the Issuer control of the smart card platform.

1.6 Issuer Control

Most smart cards are given to customers by Issuers such as banks, mobile network operators, government and transport companies. Usually in the fine print of an agreement it will say that the card still belongs to the Issuer, even if the customer has parted with some money to obtain it. The reason for this is that the cards are important to the Issuers both from a business and security point of view, and so the Issuers want to retain management rights, e.g. decide what data and functionality is offered. In that respect the smart card is very different to a PC on which the customer can usually

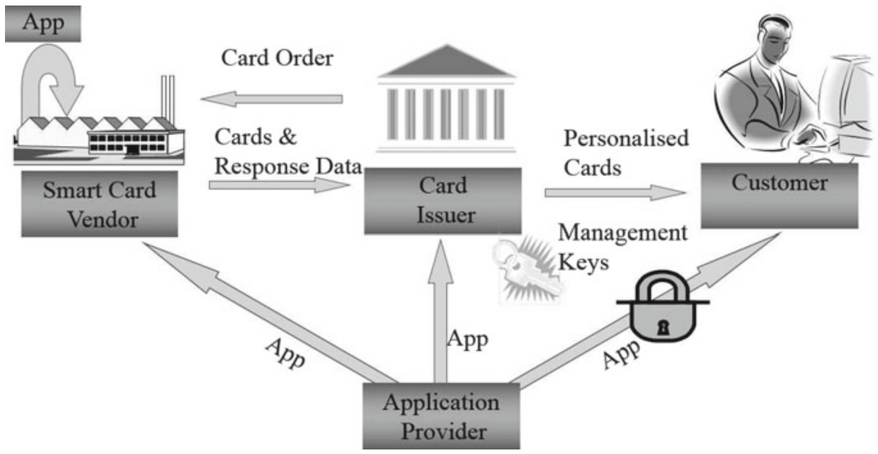


Fig. 1.9 Smart card platform management

download any data and applications that he/she chooses. If we look at Fig. 1.9 we can see where card contents are typically created and/or modified.

Consider Fig. 1.9 and the case of putting a value added application onto a smart card. Historically the *Card Issuer* would give a specification to the various *Smart Card Vendors* who would then implement the functionality in a proprietary manner and deliver this only in newly ordered batches of cards. These days card implementations need not be so proprietary (e.g. Java Card [17] or MULTOS [18]) and there is the capability to load new data and functionality after manufacture (e.g. directly or via GlobalPlatform [19]) so the situation is more flexible. The *Issuer* or its sub-contractor could personalise the cards and load appropriate data and applications not only prior to issue to the *Customer*, but could also load/manage the card contents after issue to the *Customer*. It is therefore technically feasible for third party *Application Providers* to develop card applications and offer them direct to customers, however in practice this is extremely rare. Basically all the functionality that permits the card to be managed must also be secure to prevent abuse from attackers. The secure protection is provided in the form of cryptographic functions that rely on secret *Management Keys*. Card management is therefore only possible for the *Issuer* who holds onto these secret keys. The management functionality is quite flexible and supports the idea of multiple security domains on a smart card with delegated management, although it is unlikely to be used in practice. An *Issuer* would likely argue that its control is a positive thing as it ensures tight management of the card contents and behaviours and thereby maintains security; however, there is a risk that frustrated application developers will implement on alternative and more open devices, in order to give customers the services that they desire.

1.7 Current Applications for Smart Cards

Smart cards are used in many current and real-world systems and are proposed for many future applications. In fact the capability and numbers of cards are growing rapidly in just about all areas of use. Some of the most notable applications include

- Mobile Communications
- Payment/Banking
- Transport
- Government Identity Cards/Passports
- Entitlement Cards/Health Cards
- Physical Access Control
- IT access control
- Satellite TV

For mobile communications the focus has long been on SIM cards; however, a smart phone with Near Field Communications (NFC) capability may incorporate a hardware Secure Element (SE). The embedded SE (eSE) is very much like an attack-resistant smart card chip that could be found in a SIM, and indeed its presence may eventually challenge the requirement for a removable operator SIM card. Computers and perhaps phones may also include a Trusted Platform Module (TPM), designed to give assurance in the correct state of the computing platform and software. The TPM chip has a very specific role and so it is not as flexible as a smart card chip; however, the attack-resistant properties are very similar.

Figure 1.10 shows an estimation (original source Infineon 2014) for the secure microcontroller chip annual market, including both smart card applications and

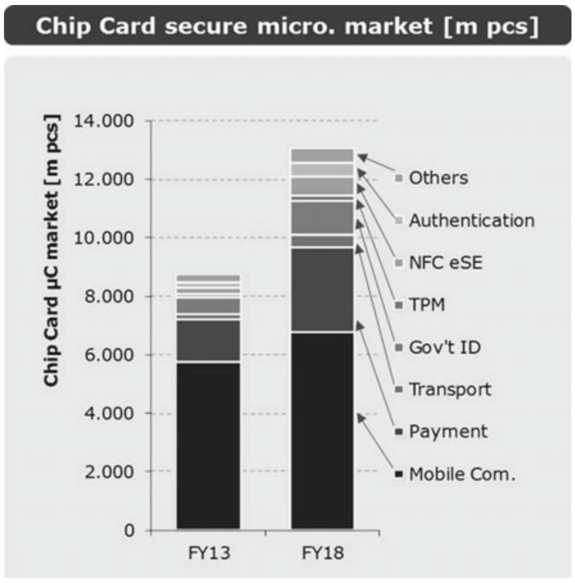


Fig. 1.10 Market snapshot for secure microcontrollers

embedded usage. The total market volume is enormous and still growing, and was predicted to rise from around 9 billion in 2013 to approximately 13 billion in 2018. These figures are even more remarkable when it is realised that they *do not* include the simpler types of RFID, and that the forecasts did not address potential demand due to the drive towards the Internet of Things (IoT). Referring to Fig. 1.10, we see that mobile communications still dominates the market in the form of the GSM [20] SIM card and the 3G/UMTS [21] equivalent USIM card (we use the term SIM to refer to both). Although there is always discussion around SIM cards being abandoned for embedded chips, it has not happened in a big way yet, and Mobile Network Operators (MNO) are resisting the change. At the time of writing there are over 4.5 billion mobile phone subscribers and as SIMs have become throw-away items it is not surprising that around 6 billion SIMs are being issued each year. Despite their ubiquity and reducing lifetime, SIMs tend to be amongst the most technically advanced cards in use, which is in contrast to the very simple call-credit phone cards that first appeared in fixed phone networks. After communications, payment is still in second place and its volumes have grown as the EMV chip and PIN standard has gained momentum around the world. The 3 billion estimates for 2018 might even be an underestimate considering that EMV is now establishing itself in the USA. Smart phones with NFC may eat into the touch and pay (PIN-less) EMV card transactions, but this is still quite new and users may have conventional cards even if they use their mobiles. Previous attempts to use mobiles for conventional mass-market payments have not always gone well, not because the technology is unsuitable, but because users struggle to configure, operate and manage the facility, especially when they wish to change phone type, bank or network. The use of transport smart cards is growing, although many are security protected memory cards, rather than secure microcontrollers. In the UK volumes have been driven by the very successful London Oyster card system [4], although the system now accepts EMV cards too and so the number of transport-specific cards issued in London could dwindle in the coming years.

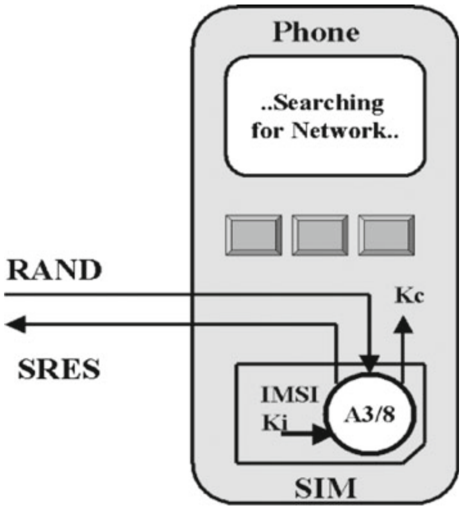
The numbers of identity cards and passports with chips are growing steadily, and in the future national security concerns will place even greater importance on reliable proof of identity. Proving the integrity of computer platforms is also a major concern which should be aided by TPM chips. However, their numbers are not showing much growth; and in fact many of the issued chips have not be enabled for use. This is not the fault of TPM technology, but rather the cumbersome process to enable it, requiring the user to take ownership of the chip. The process steps arose to protect the user against a computer or OS provider taking unauthorised control. This user protection was a laudable intention, but in reality it has largely destroyed the usefulness of the TPM and so where platform protection does exist it is often handled by proprietary secret techniques. The number of NFC eSE chips is growing. NFC when used for card emulation has had a faltering and long drawn out start, but we are now seeing serious and potentially mass market services such as Apple Pay and Android Pay. The Apple variant, at least, makes use of a hardware SE, whereas the Android version may be making use of Host Card Emulation (HCE). The latter emulates the card in software on the main CPU of the phone, so has less all round attack resistance than the hardware SE, but that does not necessarily mean that the overall solution has insufficient security protection.

In terms of authentication, there is also a lot of interest in entitlement cards and health cards and such systems may entail a roll out to every citizen within the service area, although national and internationally standardised and compatible systems seem some way off. Physical and IT access proliferate, but are not well standardised and proprietary solutions, still survive. There are many people watching satellite TV and quite a lot trying to hack the security. The smart cards are doing a reasonable job at defending the TV systems, but the future is not very clear as it seems there are all sorts of ways of delivering TV and other valued content to consumers and their many interface devices.

1.7.1 Mobile Telephony

Chapter 4 will provide an in-depth study of the ubiquitous SIM/USIM devices found in modern mobile phones; however, no introduction would be complete without taking a quick look at the most widespread and successful smart card application of all time. It is particularly useful to understand why there are smart cards in mobile phones, especially when they are not really used like physical cards at all. To find the answer requires a little trip back into history and before the GSM digital phone standard was introduced. In the UK for example there was an analog mobile phone system in use, known as Total Access Communications System (TACS for short). Being an early pioneering system, the design emphasis was on simply making the radio/communications systems work. The fact that this was achieved with the technology of the time was an astonishing achievement, but it meant that other issues such as privacy and security were rather primitive. Essentially a mobile phone held two identifiers used to authenticate the phone/user to the network. One was the normal telephone number (MSISDN) and the other was a unique electronic serial number for the handset (ESN). When a user wanted to make a call there was a signalling exchange involving the two identifiers and if the network judged the parameters to be correct and appropriately paired, then this was a valid (authenticated) user. Unfortunately that is where the security stopped and so far "confidentiality" has not been mentioned. In fact with a radio receiver an eavesdropper could listen in to any call as there was no encryption. It was also possible to identify calling parties as the MSISDN was transmitted. Clearly, this was very bad from a confidentiality and privacy perspective and it lead to some embarrassing incidents that were bad for the industry. However, the worst was yet to come as the eavesdropper was also able to receive the ESN. In theory this should have offered no advantage as the handsets were supposedly designed to prevent unauthorised re-programming of the ESN/MISIDN pairs. Unfortunately the handset security proved weak and so it was possible to create "clones", i.e. phones with the MSISDN and ESN from legitimate accounts. The first sign of cloning was when a user received a huge monthly bill for call usage. In summary, there was a major problem coupled with a lack of confidence in handset

Fig. 1.11 Phone and its SIM



security and so the mobile operators decided that in the next generation of phones (GSM) they would embed a security module in the form of the SIM card. Figure 1.11 shows the phone with its embedded SIM. The SIM incorporates a number of features that were meant to overcome the problems of the earlier analog systems, (see Chap. 4 for a more detailed description). First, it holds a customer (really card) ID called an International Mobile Subscriber Identity (IMSI). The IMSI is mapped to the real telephone number back in the network, making it harder to identify who is making a call; in fact there is also a Temporary IMSI (TMSI) that helps to further disguise the users identity. The SIM also hosts two algorithms and a secret key used for symmetric key cryptography. The naming of the algorithms is not very inspiring, but related to the candidate algorithm frameworks considered by the standards committees. Algorithm *A3* is used for authentication and *A8* is used to support ciphering. The mobile network operator has a server (called an Authentication Centre AuC) that has copies of all the card keys as well as the algorithms. Note that a network operator is free to design and use their own proprietary A3/A8 algorithms. The operation is quite simple. The AuC generates a challenge in the form of a random number RAND. It feeds this into its copy of A3 and works out the correct result for the particular card that is being authenticated. The challenge is sent to the SIM via the phone and the subsequent result SRES is then returned and compared with the expected result to decide if the SIM is authenticated. In parallel with this the A8 algorithm calculates a cipher key which is then used by the phone (A5algorithm*A5* algorithm) and the network to cipher subsequent communications. Whilst there have been a few avoidable problems (COMP128-1 algorithm [22]) the approach taken with GSM has been remarkably successful and the 3G USIM has now taken this even further by eliminating some potential weaknesses in the 2G solution (see Chap. 4).

1.7.2 Banking

In the last section we discussed how a major security and business problem justified the use of a smart card in mobile communications and now we see it is a similar story for credit and debit cards. In fact with bank cards you can not only improve security, but add new functionality in the form of off-line transactions supported by public key cryptography. The rationale for introducing the smart card and in particular the EMV smart card was the widespread fraud from using magnetic stripe cards. The reader is referred to Chap. 5 that explores banking cards and transactions in great detail.

1.7.3 Transport

The use of smart cards in transport is becoming a growth area because it offers flexibility, fraud reduction, speed and simply because users like it. The Transport for London Oyster card [4] is a prime example allowing customers to use public transport within London without the need to carry cash, queue for tickets or worry about getting the cheapest fare. There are similar systems around the world (e.g. Hong Kong [23]) and there has been interest in the use of multi-functional smart cards (e.g. credit card plus e-ticket) and mobile phones with contactless card interfaces (NFC) to make these systems even more convenient and popular.

It is worth noting that the challenges for a transport card system are a little different to say a mobile communications or financial transaction. Transport card transactions need to be fast because you cannot afford huge bottlenecks of customers at train stations or when entering a bus. This normally means that you cannot use an extra PIN code as you might with an EMV card and so we are restricted to single factor authentication (something you have). This is an operational rather than technical limitation, but it probably means you would wish to restrict the maximum value transacted in this manner. There are technical challenges in how to get enough power through the contactless interface to work rapidly. In fact there is always a very sensitive trade-off between the transaction speed, security and cost of the card. Perhaps the biggest challenge to transport systems comes from standardisation of the e-ticket. Customers would like to have one ticket that they can use for all travel nationally and ideally internationally. The use of contactless EMV cards is gaining ground in metropolitan systems, but has yet to be adopted for long-distance and expensive fare journeys. In the UK, the ITSO [24] organisation has been developing supporting national standards and requirements; however, at the time of writing the most successful UK system (Oyster [4]) is not an ITSO system, although the London card readers are expected to be compatible with ITSO standard cards.

1.7.4 Identity and Passports

As more and more government and regional systems move to electronic processes and transactions, it becomes necessary for citizens to have some kind of compatible electronic identity. For example, the modern passport includes a smart chip which can be accessed as part of normal travel processes. Many countries have national ID cards, some of which are based on smart card chips; however, the introduction of new cards can be emotive for countries that are not used to them. Some years back there was a planned scheme for the UK that sparked controversy, partly because of the biometric information to be stored on the card, but also because of the large amount of personal information that would sit on central databases and be accessible to numerous organisations. In the end the project was cancelled largely due to cost issues. The end user cost of passports and/or ID cards is often at least an order of magnitude greater than the best smart cards used in banking and mobile communications and so there is more than enough scope to include a technically advanced and secured chip. Perhaps the biggest problem is that the card/chip attracts all the attention, whereas much of the security challenge and indeed cost will be in the rest of the system and associated processes.

1.7.5 Entitlement and Health

Entitlement and health cards are perhaps good examples of smart cards that prove an identity and associate with it some private data and rights, but then so too are SIMs, EMV cards, IDs and passports. Clearly there are many reasons for a citizen to be able to identify themselves electronically and whilst smart card chips have the technical capabilities to satisfy the requirements of multiple systems and services, it appears that we will be burdened with more and more smart cards for the foreseeable future. Considering the entitlement card, how might it differ from say a passport? Well entitlement cards are likely to be issued regionally, perhaps for low-value privileges such as library access or discounted transportation. As they are part of public service delivery and often aimed at disadvantaged citizens, the cards are likely to be free issued. These factors taken together suggest that the smart card chips might be low cost with limited security and that systems from different national and international regions may not be standardised and/or compatible. Health cards by comparison are likely to have similar cost pressures, but are more likely to be standardised at least nationally. Data privacy is the major factor for any system that is used to secure health records, as misuse of this information could lead to discrimination and could damage an individual's relationships and reputation. It is therefore vital that a health smart card is not only very secure (despite the cost pressures), but that it underpins a complete system that has been rigorously evaluated for the protection of citizen's private health information.

1.7.6 Physical and IT Access Control

It is quite common for employees to be given an ID card to gain access to their place of work. Unfortunately, the cards are often not very smart, e.g. either the simplest of contactless smart cards or perhaps magnetic stripe cards. Where a magnetic stripe card has been replaced by a contactless card it may have had more to do with a desire to improve reliability than security. In fact the very simplest contactless cards are the classic RFIDs. They present an ID by radio means with no attempt at adding security or disguising the transmitted ID. As an RFID may be remotely eavesdropped (sniffed) you might even argue that the simplest cards are less secure than magnetic stripe as you at least need possession of the latter to read it! The real reason that these cards are used is that the companies consider that a sophisticated attack will not be used to gain access, basically because there are much easier ways. Cards often get left at home, lost or destroyed by various means and it is an operational challenge to know if every card in the company database exists (or not) and whether they are all still in the hands of authorised employees. Stealing a card is the easiest way to gain access, but you can raise the security bar a little by requiring employees to also enter a PIN code, although this impacts on the convenience and the flow through access points, as well as the operational management of PIN codes for forgetful humans. It is interesting to note that IT access control can be more sophisticated than physical access control especially as physical access might allow a criminal to steal all the company assets including the IT systems! Smart cards for IT access are issued for good security reasons and normally with the support of the IT department who are well used to managing user accounts, passwords and PINs. IT hardware and software is generally expensive and so the cost of the smart cards readers, whilst not negligible, is not a huge cost impact and so it should be possible to buy cards with reasonable capabilities. Normally these would be compatible with IT defacto standards, e.g. for access to Microsoft Windows.

1.7.7 Satellite TV

Whilst commerce used to centre around the exchange of physical items, today's customers are keen to buy less-tangible goods that have no less perceived value. In the digital multi-media domain we may want to watch something or listen to something for our entertainment and therefore have to pay for the right. The control of this is generally called Digital Rights Management (DRM) [25] and a good example is satellite TV where a customer can decode a transmitted programme, such as a football game, if he/she is in possession of a set-top-box receiver with appropriate security functionality and rights. Satellite TV is very desirable and not particularly cheap and so is a prime target for hackers. Defending the rights is not trivial as this is a broadcast media and so there has to be some global secret elements. Various security solutions exist [26–28] (see Chap. 6 for a detailed discussion) that can be crudely grouped into smart card-based and non-smart card-based systems. The defenders of satellite TV systems have a realistic outlook on life, i.e. they expect to get attacked, to resist for a while, to update a few times to fix weaknesses and finally to roll out a new solution. This outlook is why the card-based solutions exist, because it is much cheaper to issue a new smart card to improve security than a whole set-top box. It is also a convenient way to adapt and configure a general set-top box to different security/content providers. Not a great deal is published about the cards used in satellite TV systems (aside from hacker forums) and the arguments against security by obscurity cut little ice with the security system providers. Everything is done to make the hackers' task as difficult as possible and so cards can be completely non-standard and armed with a few hidden tricks and surprises. Although cards and Set-Top Boxes (STB) have done their jobs at enforcing access rights to satellite-delivered content, the solution is looking rather dated. Television is now delivered over the Internet to PCs, phones, tablets and smart TVs, none of which are that likely to support a plug-in smart card from the satellite service provider. More flexible DRM solutions are required to match these new delivery channels and devices, and so the days of the DRM smart card may be numbered.

1.8 Smart Card Application Development

From the foregoing text it can be seen that the smart card is a secure microcontroller that has been successfully used in a wide variety of applications. There are in fact many more applications including e-purses, lottery, voting, loyalty, user menus, games, etc., but all of them had to be developed at some stage. An important consideration is therefore how to implement a smart card application and whilst this should not be beyond the capabilities of most programmers there are quite a number of ways to go about this, especially in the case of SIM/USIM cards. The obvious starting point is that a microcontroller can be programmed in a form of assembly code or perhaps via a *C* compiler. You can easily buy smart cards for this, but they tend not to be secure, i.e. they are not designed for attack resistance. Of course the secure chip manufacturers and smart card vendors have developed secure devices, but you would normally be prevented from accessing the very low-level code and have to content yourself with developing above the operating system and via some API or toolkit/interpreter. There are good security reasons for this and the API interfaces offer a lot of convenience to the programmer at the expense of speed. In the SIM world, one of the earliest programming facilities was known as a SIM Toolkit Scripting. The SIM Toolkit functionality was originally described in GSM 11.14 [29] and was implemented in the form of a primitive scripting language which could be understood by a script interpreter on the SIM card. The most common and successful use of this was for custom service menus that appeared (to the user) to be stored in the handset, but where actually hosted and managed by the SIM. Earlier implementations were not flexible and so it was difficult to correct or change the Menu services although this improved over time. The biggest problem was that the scripting languages were vendor proprietary and so a network operator would have to implement/test the *same* functionality multiple times. Bearing in mind that the vendors used different development tools and that testing with many handsets requires a huge effort, the problem should not be underestimated.

As a solution, another development route was created which in principle offered many advantages over SIM Toolkit scripting. The idea was to implement a very simple Menu browser on the SIM so that it provided simple menu options and handled the user or network responses. There were a variety of browsers including the WIB (Wireless Internet Browser [30] from SmartTrust) and the S@T Browser (SIMAlliance Browser). The clever part was that the SIM services sat on a network server rather than in the SIM itself, so providing you did a good job of testing the browser implementations from the various card suppliers, you could implement new SIM-controlled services without changing its stored functionality or necessarily re-testing the SIM. In practice some re-testing was advisable, but all the changes were really in the network. This all held great promised until you realise that an SMS bearer was used, so when you selected a menu option an SMS message had to be sent to the server and a response message returned, also via SMS. At the height of the browser development, the SMS service was quite unreliable and transmission might take a few seconds, but could also take a minute. Measures to get a faster turnaround of SMS helped the situation and some operators went for a menu caching approach, but that really made it a conveniently managed set of SIM hosted services rather than the true browser approach. Another problem was that although the browser approach removed some of the problems from card vendor proprietary systems it risked bringing in a proprietary and single-vendor system component. The popular WIB needed a Wireless Internet Gateway (WIG) in order to be useful and whilst the WIB was free the WIG represented a considerable investment and potential dependency on a single supplier. This is one of the reasons why the Alliance of SIM vendors produced the S@T browser. Although the SIM browser seems to be rather side-lined as a development method, the idea was pretty good and faster communications plus a more open-source approach to the gateway might see a resurgence. The odds may not be great as in the meantime developers have found their favourite platform in the form of the Java Card [17].

Java is popular as it abstracts the programming environment from the underlying chip platform, which means that applications should in theory run unmodified on Java Cards supplied by different card vendors and using different chips. For a long time this was far from reality and even today it is wise to repeat testing for all card types. Of course someone still has to develop the low-level code to provide an operating system plus the Java virtual machine/run-time environment, but that is just done once and usually by the chip or card vendor. Flexibility and card management is provided by the GlobalPlatform [19] functionality which helps to support secure application and data loading, modification and quite sophisticated security domains and channels for isolating multiple applications. Java Card still suffers from the fact that the functionality has to be developed and loaded on the card itself (rather than a remote server), which in some applications creates testing and card management issues. Java Card and GlobalPlatform are described in detail within Chap. 3 along with another multiapplication card that was in use whilst Java Card was in its earlier stages and that some would claim is more secure. Given that smart cards are usually used for their security attributes, it might therefore seem a little odd that MULTOS [18] was not the developers favourite in place of Java Card, especially as Java is one of the languages that can be used for MULTOS development. The reason is partly due to the fact that MULTOS was designed with the highest standards of security in mind and that meant the whole development and application processes were very controlled, requiring various approvals/certifications and accompanying paperwork before a developer could get his/her application approved and loaded onto a card. This seemed to deter developers and as the vast majority of smart cards were (and still are) for mobile networks that did not insist on formal security evaluations, a lot of the activity headed into the Java camp, which also meant that more freely available tools became available to ease development. MULTOS has not disappeared; however, and these days it is easy to develop with, and should still be given consideration particularly for very high security applications. There have also been recent moves to apply MULTOS to IoT security, exploiting the fact that personalisation and application loading is based on public key (rather than shared secret key) cryptography, making it easier to perform in untrusted and off-line environments.

1.9 Development, Roll Out and Lifecycle Management Issues

Developing software for smart cards is in many respects like any other software development, but if you make a mistake it can be a very big one that can haunt you for a long time. When you design the data content and functionality of a smart card you should capture all current and foreseeable future requirements, which is almost an impossible task as no one can accurately predict the future. If you are smart, you design in some flexibility to enable changes and to add more data and functionality in future. Unfortunately, this will likely be resisted by the purchasing department who believe in saving pennies today rather than the promise of rich, yet unspecified new services in the future. Whatever the final agreed compromise, it is translated to a smart card profile that is a definition of how the chosen smart card should be configured. Depending on the application, there may be a great many profiles and reader combinations in use. For example you could have 100 SIM profiles and 1000 s of phones in your network. Testing is really important and to really understand this, consider how much money you could lose your company if a bug slips through. Let us say we have a new mass-market commercial card that we think is properly tested and a big order of 1 million cards is needed, which for simplicity we will say costs £1 million. If you missed a serious bug you may need to recall and replace the cards, a process that is known to cost an order of magnitude more than the cards, i.e. £10 million, because it involves customer care and communication as well as the replacement card and its delivery. Indeed, the cost could be higher if disgruntled customers chose an alternative service provider. This is one of the reasons you build in remote management, but changing 1 million cards would still be a major undertaking.

You might decide to live with a minor bug and so then the interest will be on the normal card lifecycle. For bank cards this is defined as a few years, but for other cards, e.g. SIM there are no expiry dates. Whilst a SIM might be discarded after a month it is not impossible to find SIMs in use that are over 10 years old. This comes to another important point regarding new service roll out. A company will want a great new service to reach all its customers instantly; however, card-based applications can rarely offer this. If the service requires a new form of card then on the launch date you will have zero customers. If you wait for the cards to expire or wear-out then you may wait many years and if you swap customer cards you know it will be expensive. This sounds like some unfortunate bad-luck situation, but often this legacy problem was actually designed in because of the catch-22 (conflicting logic) of smart card deployment. That is, you need spare capacity and perhaps the most advanced capabilities of the smart card for important services that are not identified when the card is designed, whereas the cost of the card is only justified by the applications that are known to be essential at design time. The situation is not helped by the fact that a marketing strategy or service plan is usually much shorter duration than the life of the card. Another way to get things wrong is to succeed in providing all the necessary and forward thinking card capabilities only to find that the envisaged local/remote management platform is deficient or has simply became a budget cut. One must always remember that a smart card is a sophisticated, personalised and managed computer platform that is vital to a users secure use of a system or service. With proper design and supporting management systems it can be used for many years. Over-specifying a smart card from the bare minimum has a very tangible cost and although it may only be a few pennies or cents per card this starts to become significant for large deployments. However, the true cost of issuing minimum specification devices is less simple to determine as it may be the denial of a new service to a customer, a reduced card lifetime (and earlier replacement cost), a poor service or perhaps the loss of the customer to a competitor.

1.10 In Conclusion

This chapter has attempted to provide an introduction to a very wide range of smart card-related issues, which has hopefully been a good starting point for newcomers to the field and those that have perhaps previously focussed on one business or technical area. Of course only an overview has been possible here, but much more detail can be found in the following chapters. A few words of wisdom might be useful to finally conclude this introduction.

- Smart cards are primarily used because they are tamper (attack)-resistant security tokens.
- They are often personalised and managed computer platforms that can be in operation for many years.
- They are not magic devices that make a system secure when it otherwise has bad design, implementation, algorithms, keys and processes.
- They are always part of a system solution, and they tend to be the simplest part.

Acknowledgements The author wishes to thank Vodafone, Giesecke and Devrient, Transport for London, The UK Cards Association, Orange Labs (UK), Visa and ITSO, plus all the ISG Smart Card Centre industry supporters for their encouragement and support over many years.

References

1. CEN TC 224 WG15, European Citizen Card, 2007. More Information Available via http://ec.europa.eu/idabc/servlets/Doc59a8.pdf?id=28716, cited 09 Apr 2016.
2. International Civil Aviation Organisation (ICA0) Doc 9303, 7th Edition, 2015. More Information Available via http://www.icao.int/publications/pages/publication.aspx?docnum=9303, cited 09 Apr 2016.
3. EMV Books 1-4, Version 4.3, Mov 2011. More Information Available via https://www.emvco.com/specifications.aspx?id=223, cited 09 Apr 2016.
4. Transport for London Oyster Card. More Information Available via https://oyster.tfl.gov.uk/oyster/entry.do, Cited 09 Apr 2016.
5. HID Corp Technology Basics Whitepaper Understanding Card Data Formats, 2006. More Information Available via https://www.hidglobal.com/sites/default/files/hid-understanding_card_data_formats-wp-en.pdf, cited 09 Apr 2016.
6. W. Rankl and W. Effing - Smart card handbook, 4th edition, John Wiley, 2010.
7. Card Watch "Types of Card Fraud". More Information Available via http://www.cardwatch.org.uk/, cited 09 Apr 2016.
8. Financial Fraud Action UK, Fraud The Facts, The definitive overview of payment industry fraud and measures to prevent it, 2013. More Information Available via http://www.theukcardsassociation.org.uk/wm_documents/3533.
9. International Organization for Standardization, ISO/IEC 7816 1-4 Identification cards - Integrated circuit cards - Cards with contacts, 2011.
10. ECMA (Standard ECMA-340) Near Field Communication Interface and Protocol NFCIP-1, 3rd Edition, Jun 2013. More Information Available via http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-340.pdf, cited 09 Apr 2016.
11. Anderson, R. and Kuhn, M., *Tamper Resistance - a Cautionary Note*, In the Second USENIX Workshop on Electronic Commerce Proceedings (pp. 1–11), 1996.
12. International Organization for Standardization, ISO/IEC 14443 Identification cards - Contactless integrated circuit(s) cards - Proximity cards, 2016.
13. MIFARE. More Information Available via https://www.mifare.net/en/, cited 09 Apr 2016.
14. Auguste Kerckhoffs, "La cryptographie militaire", Journal des sciences militaires, vol. IX, pp. 5–83, Jan. 1883, pp. 161–191, Feb. 1883.
15. K. Nohl, H. Plotz, Little Security Despite Obscurity, presentation on the 24th Congress of the Chaos Computer Club in Berlin (Dec 2007).
16. Mayes K and Markantonakis K On the potential of high density smart cards, Elseivier Information Security Technical Report Vol 11 No 3, 2006.
17. Oracle, 2011, Java Card Classic Platform Specification 3.0.4. More Information Available via http://www.oracle.com/technetwork/java/javacard/specs-jsp-136430.html, cited 09 Apr 2016.

18. MULTOS website. More Information Available via https://www.multos.com/, cited 09 Apr 2016.
19. GlobalPlatform, GlobalPlatform Card Specification v2.3, Oct 2015. More Information Available via http://www.globalplatform.org/, cited 09 Apr 2016.
20. M. Mouly, M-B Pautet, The GSM System for Mobile Communications, Cell & Sys. Correspondence, 1992.
21. Friedhelm Hillebrand, *GSM & UMTS - The Creation of Global Mobile Communication* - Wiley, 2002. ISBN: 978-0-470-84322-2.
22. COMP128-1 attack. More Information Available via http://www.isaac.cs.berkeley.edu/isaac/gsm-faq.html, cited 09 Apr 2016.
23. Octopus. More Information Available via http://www.hong-kong-travel.org/Octopus/, cited 09 Apr 2016.
24. ITSO, Specification v2.1.4, 2015. More Information Available via http://www.itso.org.uk, cited 09 Apr 2016.
25. Wikipedia, Data Rights Management, 2016. More Information Available via https://en.wikipedia.org/wiki/Digital_rights_management, cited 09 Apr 2016.
26. Canal+ website. More Information Available via http://www.canalplusgroupe.com/, cited 09 Apr 2016.
27. Irdeto website. More Information Available via http://www.irdeto.com, cited 09 Apr 2016.
28. NDS (now Cisco) website. More Information Available via http://www.cisco.com/c/en/us/solutions/service-provider/service-provider-video-solutions/index.html, Cited 09 Apr 2016.
29. 3GPP, TS 11.14 V8.18.0 Specification of the SIM Application Toolkit for the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface (Release 1999) version 8.18.0, Jun 2007. https://www.3gpp.org/, cited 09 Apr 2016.
30. Gisecke and Devrient, Smart Trust WIB, 2011. More Information Available via https://www.gi-de.com/gd_media/media/en/documents/brochures/mobile_security_2/smarttrust_1/SmartTrust_Wib.pdf, cited 09 Apr 2016.

Chapter 2
Smart Card Production Environment

An Overview to Smart Card Production Processes

Claus Ebner and Thomas Goetz

Abstract This chapter gives an introduction to the production steps in the lifecycle of a (smart) card. After a short introduction, the manufacturing of the card body will be described. The next paragraphs give information on the personalisation process chain from data processing and on to card personalisation and additional services such as packaging and shipment. A separate paragraph focuses on quality and security issues. At the end, there are a few thoughts on current trends and challenges for the smart card industry.

Keywords Card body production · Smart card personalisation

2.1 Introduction

There are two main ways to distinguish card types. On the one hand, it is based on the related application/issuer type; on the other, it is the technical features and/or physical characteristics. As there is a close relation between the two—e.g. an ID card for government bearing security features in the card body—this chapter will focus on the "application view".

Smart card-based applications are developed as a pivotal part of the solutions for securing the mobile life. The areas in scope are grouped into Financial Institutions, Telecom Industries and the Public/Industrial Sector.

2.1.1 Smart Cards for Financial Institutions

The probably most traditional area and use case for smart cards is the well-known banking cards issued by Financial Institutions for securing financial transactions.

C. Ebner · T. Goetz (✉)
Giesecke & Devrient, Munich, Germany
e-mail: thomas.goetz@gi-de.com

C. Ebner
e-mail: claus.ebner@gi-de.com

K. Mayes and K. Markantonakis (eds.), *Smart Cards, Tokens, Security and Applications*, DOI 10.1007/978-3-319-50500-8_2

Table 2.1 Smart card sizes

	Standard	Height (mm)	Width (mm)	Area (mm)
ID-1 Card		53.98	85.6	4621
Plug-In	1989	15	25	375
Micro-SIM	2004	12	15	180
Nano-SIM	2012	8.80	12.3	108
MFF2	2009	5	6	30

Applications range from traditional credit (pay later), debit (pay now) or stored value cards (pay as deposit) to additional applications for authentication, loyalty and transit. The standards and specifications are defined by the global schemes Visa and MasterCard or by local payment schemes in many countries of the world.

In banking, there are the standard debit and credit cards in ID-1 format (see Table 2.1)—both with similar characteristics: a multilayer (usually four–five layers of individual plastic foils) card body with printed design, some optional printed security features, a magnetic stripe, a signature panel, a hologram and with a chip. The chip module embedded in the card body may offer a contact-based interface, a contactless interface using an embedded antenna or a combination of both interfaces. In most of the mature markets, dual interface cards are considered as standard. The operating system can be native, Java or based on local specifications.

New variations include non-standard ISO/IEC7810 cards in smaller sizes (e.g. VISA mini) or different shapes (e.g. MasterCard MC^2). With the evolving trend to contactless payment even other form factors have shown up or chip modules embedded in the shell of a mobile phone, in a sticker, in a key fob, token or any other type of wearable device.

The screen and offset printing of card artworks makes use of special colours and may create special visual effects. The design of card bodies, their layers and corresponding specialised production techniques are applied in order to deliver the requested features: Translucent, Perspective, Coloured Edge, Tactile, Nacre, Fragrance, Metal Effect, Durocore, Poly Vinyl Chloride (PVC) recycled, Poly Lactic Acid (PLA)—a PVC-free card, Shapes, CDplus Connectcard.

The card appearance is more and more seen as an opportunity for the card issuers to differentiate against competition. There is a very strong demand by the banks to get their card "on top of wallet". An enhanced user experience may increase consumers' loyalty to their bank.

The default for optical personalisation of the card is either by embossing or by laser engraving. Optionally, this can be combined with or even replaced by thermal transfer, thermal sublimation, re-transfer or inkjet.

Shipment of personalised cards to the card holder is performed by national mail services or by courier. Service bureaus produce the individual mail pieces including welcome letter, the card or companion packages of cards and additional marketing material. For premium segments, special packaging in high-value wooden or leather boxes is not unusual.

2.1.2 *Smart Cards for Telecom Industries*

The deployment of smart cards as secure element for securing mobile connectivity has meanwhile reached the largest volume of smart cards issued on a global scale. Subscriber Identity Module (SIM) card-based applications authenticate the subscriber against the network. The SIM serves as a secure platform:

- For additional services like Short Message Service (SMS), phone book, mobile banking
- For multi-applications like Near-Field Communication (NFC) for Payment, Ticketing, Access control
- As network node for machine-to-machine communication, secure storage, digital identity, Long Term Evolution (LTE) mobile network technology

The SIM card as secure element is offering interoperability in the environment of a great variety of handsets and other connected devices increasingly in the area of machine-to-machine communication. Here, the automotive sector is an important field where secure connectivity will play a very important role in the future.

The SIM card allows the Mobile Network Operator (MNO) full control over the offered services independent from the handset manufacturer or other third parties. Services on the SIM card are in the control of the MNO. SIM cards support the range from simple feature phones up to complex and high-end applications for smartphones. The MNO-centric services comprise roaming services, device and application management, quality of service information. Technology on SIM cards range from Java, Wireless Internet Browser (Wib) and SIMalliance Toolbox (S@T).

The card body may be either multilayer or injection moulded—with a decreasing trend for multilayer. For cards either with a short life cycle, or only serving as carrier for the plug-in module until mounted in the mobile, usually the cheaper variant is chosen.

For a card body which has no security elements, optical personalisation is either done by inkjet and thermal transfer printing or by laser engraving.

Mobile phones which take a complete ID-1 card are long gone, but even the ISO/IEC 7810 ID-000 Plug-in size has already smaller successors: the Micro-SIM, the 3rd Form Factor (3FF) followed by the Nano-SIM or 4th Form Factor (4FF). All of the above represent removable (pluggable) secure elements. For direct mounting on Printed Circuit Boards (PCB), the Machine Form Factor (MFF2) was introduced. This Surface Mount Device (SMD) part is soldered to the electronic board and thus, not intended for removal or exchange by the end user. It is mainly used in assemblies for communication boxes in various Machine-to-Machine (M2M) environments. Whilst 2FF and 3FF have the standard smart card thickness of 0.76 mm, the 4FF is reduced to 0.67 mm.

The multitude of different shapes leds to some logistical challenges for the MNOs. For different types of handsets, different types of SIM cards are to be used. An exchange of mobile device might call for a differently shaped SIM (Fig. 2.1).

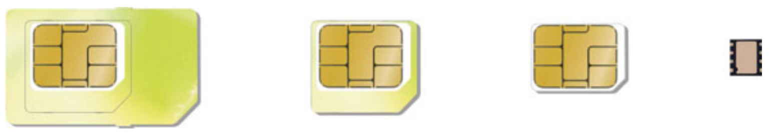
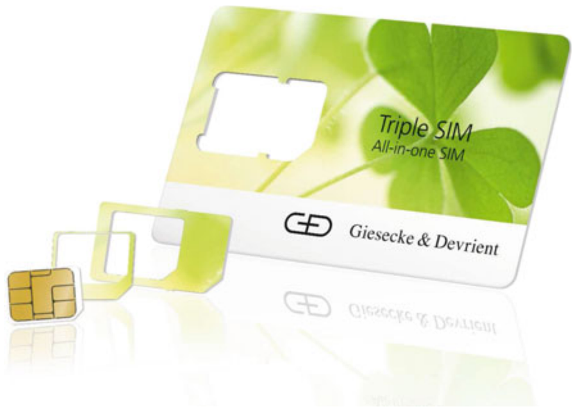


Fig. 2.1 Smart card sizes (Image Copyright: Giesecke & Devrient)

Fig. 2.2 Triple SIM (Image Copyright: Giesecke & Devrient)



In order to overcome this issue, the Triple SIM as a flexible SIM card that supports all relevant SIM forms was introduced. It supports the standard Plug-in (2FF), the Micro-SIM (3FF) and the Nano-SIM (4FF). The inherent design characteristics of such a combined card allows the Plug-in to be used in the full, unmodified 2FF size, or allows the user to "snap out" smaller sizes depending on the mobile device requirements. It also supports the reassembly of the individual smaller sizes to the next bigger size (3FF or 2FF Plug-in) for multiple times but usually only with the original card frames (Fig. 2.2).

Delivery of SIM cards is performed in various different packaging which serves the purpose of branding and marketing, positioning the MNO as trendsetter, for security reasons and for delivery of add-ons like CDs. Value-Added Packaging (VAP) can be in form of a blister, as Plug-in only, in a card board packaging, as standard lettershop mailing, in a SIM card dispenser or in single flow wrapping.

2.1.3 *Smart Cards for Public and Industrial Sector*

Both in the public and the industrial sector, the smart card is serving as secure element. It is used for securing identities, for enterprise security as well as for securing public infrastructures. The public sector comprises applications for health or social insurance as well as for ticketing mainly for transit and fares. In form of an ID card, it replaced the traditional personal documents in many countries of the world.

The highest requirements for the card body can be found for government cards and here of course especially for long-life ID cards. The card body is usually of multilayer type (up to 9), containing security features such as mentioned for payment cards plus even more sophisticated ones, e.g. a multiple laser image, micro line print, guilloches, invisible fluorescent ink print. Healthcare cards usually have a contact-based chip and most new ID cards use contactless technology.

For optical personalisation all techniques can be used—preferably laser engraving due to security reasons. For photos also colour dye sublimation or retransfer technology is used.

2.2 Smart Card Production Steps

2.2.1 Overview

On the way to the final product for the customer there are several steps in card production. First, there is the manufacturing of the card body—which includes making of the plastic, printing and adding additional elements, such as the magnetic stripe. This is followed by embedding the smart card module, which itself went through the steps of test and probably completion and initialisation.

An optical and electrical personalisation transforms the smart card to an individual one. This often is accompanied by related services, such as card carrier personalisation, mail fulfilment and packaging. The following paragraphs describe these steps in more detail.

2.2.2 Card Body Manufacturing

The card body production itself may be divided into several steps again—depending on the technologies used and the features of the card: see Fig. 2.3.

2.2.2.1 Materials

The basic material used for cards is either supplied as foil for laminating or as granulate in case of injection moulding. The classical material used is PVC, but due to environmental discussions and higher lifetime requirements as well, other materials gain importance. Table 2.2 gives a short overview summarised from [1]:

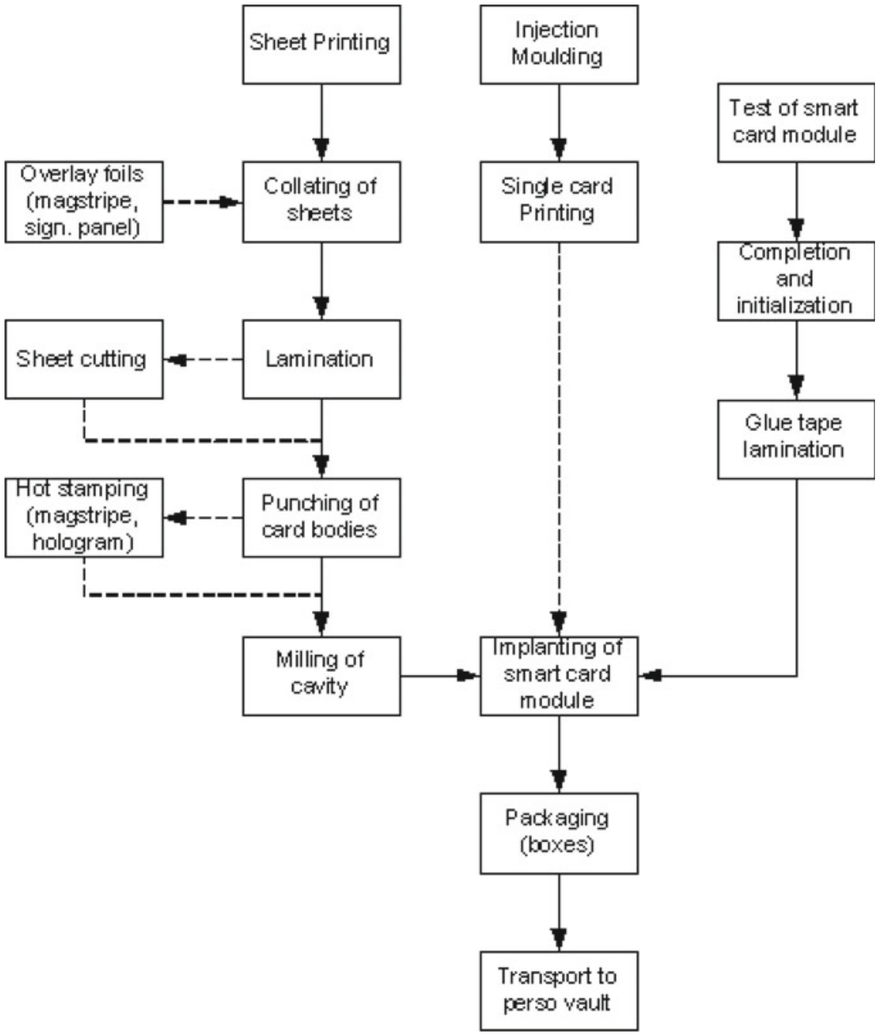


Fig. 2.3 Card body manufacturing flowchart

2.2.2.2 Printing Technologies

Offset Printing

The offset printing technique is the most common one used today in industry. It starts with the making of offset printing plates for each printing colour, usually directly from a digital source made with the computer (“Computer to plate”—CTP).

Table 2.2 Card materials (See Glossary for abbreviations)

Material	Advantages (+)/Disadvantages (-)
PVC	(+) Low price, many years of experience, recycling possible
	(-) Environmental compatibility, limited thermal stability
PC	(+) high temperature stability and mechanical strength, recycling possible
	(-) high price, low scratch resistance
ABS	(+) injection moulding suitable, temperature stability, recycling possible
	(-) does not comply with ISO standard, not classified as environmentally friendly
PETG	(+) best material regarding environmental compatibility, middle price, recycling possible
	(-) process not as easy and well known as for PVC

The image is exposed to the light sensitive plates with a laser beam. After development of the plate and chemical treatment there are zones which attract ink and others which attract water.

The plates are mounted to printing cylinders which during their rotation run against water rollers and ink rollers. The water rollers dampen the non-image parts of the plate, the ink rollers dampen the image area of the plate with ink.

The plate then transfers the ink to the rubber blanket of a second cylinder, which in turn offsets the image onto the foil running between it and an impression cylinder.

There is also a waterless variant of offset printing using special inks and UV technology. This technique is used in machines for single card printing, where the design is applied to white cards—mostly coming from an injection moulding process.

Screen Printing

The other technique used for card printing works with a porous woven fabric which is stretched over an aluminium frame. A stencil is created on the screen by filling its mesh for the negative parts of an image.

The common method to do this is the photo emulsion technique. A positive film of the image is made and placed over the screen, which is coated with a light sensitive emulsion. Exposed to ultraviolet light, the emulsion will harden in the parts of the screen, where the UV light passes through the transparent areas of the film. The non-hardened emulsion will be washed away afterwards from the screen and a negative stencil of the image will be left.

In the press, the screen is placed over the foil to be printed and filled with ink. A rubber blade (called squeegee) then is pulled across the screen which fills the holes of the mesh with ink. In a second step, a squeegee will press the ink through the mesh onto the foil which is pressed against it. The printed foil must now dry before the next colour can be applied.

Digital Printing

For high volume printing, today there is no alternative to the techniques described above. But as there is a trend to address card issuers more and more individually, small editions (even “lot size 1”) are topics for the card industry.

Digital printers working with thermal sublimation dye or retransfer printing are used to print individual designs onto white cards. Though the quality of this technique has so far not reached the level of Offset or Screen Printing, the results are already very well accepted by card issuers.

2.2.2.3 Lamination

Card bodies manufactured with the lamination technique consist of two or more foils, which are pressed together under high temperatures. As the foils are of thermoplastic material, they will establish a connection under heat when their softening temperature is reached. The most common compositions are four- and five-layer cards, for contactless and ID cards even up to nine layers are put together. No matter how many layers are used, as the physical parameters of a card are defined in ISO, the sum of the foil thicknesses has to be less than 840 microns.

To protect the design printing there are two ways: if there is external printing on the outer layers of a card the surface will be covered by a transparent varnish. In the most common case of internal printing, transparent overlay foils will be laminated over the design which provides a better resistance against scratching and abrasion.

Besides design and overlay foils there are other components which can be applied in the lamination process. Magnetic stripes and signature panels brought onto a foil before are often added in this process step already. For contactless cards a "pre-lam" inlay containing the chip module and antenna is one of the layers in the lamination process.

Before entering the lamination press, the layers have to be collated in such a way that the images for front and back side match exactly and the location of additional elements, such as magnetic stripes or contactless inlays, is within given tolerances. This may either be done by hand or using a sheet collating machine. The simplest way is to align the sheets using their ledges or using adjustment holes in the sheets. If more precision is needed, printed crosses on the foils are brought together using a special table with two cameras—one for the front and one for the back design. In any case, the foils will be stapled together by a heated spot stamp in the rim.

These pre-mounted sheets are stacked together with thin, highly polished metal plates. This stack is put between one of the several heating plate pairs in the laminator. Depending on the necessary process parameters—which are specific for each product (regarding the type of materials, etc.)—the layers are heated for a certain time and under certain pressures. After cooling down under pressure of the laminated sheet, the card bodies will be punched out in the next process step. If necessary, the sheets will be cut to fit the punching machine.

2.2.2.4 Injection Moulding

For GSM cards which mainly serve as carrier for the plug-in module, the trend is to choose the cheaper process of injection moulding. The cavity needed for the chip module is already created in this process, so that no milling is necessary afterwards.

The preferred material for injection moulding is Acrylonitrile-Butadiene-Styrene (ABS). The plastic granulate is pressed under high pressures into the pre-heated mould form. The material melted by heat and shearing forces fills the shape of the mould and solidifies. The form is opened and the work piece ejected.

The two main challenges for injection moulded cards are to find the right process parameters to cope with the shrinkage of the material and to ensure the quality of the relatively thin part under the module cavity.

Another important attribute of injection moulded cards is their printability, as the card design will be applied afterwards in a single card printing process before the chip module is implanted.

2.2.2.5 Adding Other Card Elements

Signature Panel

Everybody who has a Payment Card in his/her wallet, knows that he/she has to sign his card before he/she may use it. In order to do so with a customary ballpoint pen, a special signature panel is necessary.

Signature panels are applied with two different techniques, laminating or hot stamping. Paper signature panels are mounted to the outside overlay foil of a card and will connect to the card surface during the lamination process. Another option is to create overlay foils with a printed signature panel by the use of special colours in a screen printing process. Again, this overlay will be applied in a lamination process.

The hot-stamping technique works with prefabricated elements which are transferred from a carrier tape to the card body by the usage of a heated stamp. The elements—such as signature panels—are covered with an adhesive which activates under heat and pressure. Under the hot stamp, the element will bond to the card surface and in turn lose its connection to the carrier tape.

Magnetic Stripe

The magnetic stripe which is a main element for all payment cards, needs to be put onto the card at a certain position. The techniques used to apply it are the same as for signature panels. Either the magnetic stripe comes on an overlay foil and is laminated or a hot-stamping process is used.

Hologram

Another security element—for example known from some payment cards—is the hologram. It also is applied to the card body using a hot-stamping process.

Components for Contactless Cards

While for smart cards with contacts, the chip module will be embedded after card body production, for contactless cards this happens by lamination of inlays which contain antenna and contactless chip module.

There are three main ways to manufacture the antenna for such an inlay:

- In the wire embedding technique, the wire is laid directly onto a plastic foil and melted into it by using ultrasonics.
- Etched antennas are created using photolithography with copper covered plastic foils. The surplus copper will be etched away by acids only leaving the antenna shape on the plastic foil.
- The printed antenna can be produced using a special silver ink in a silk screen printing process.

The technology to connect the smart card module to the antenna depends on the antenna type. For embedded antennas, it is micro welding while for etched antennas it is soldering.

No matter what technologies are used to create a contactless card, it is essential that the unevenness caused by antenna and module is equalised to achieve a good card surface.

In the case of dual interface cards, there are currently two methods for establishing the electrical connection between the contact chip module and the embedded card antenna.

Method 1 introduces a physical connection between the contact pads of the dye in the module and the card antenna. This is established either by a conducting glue which is applied in the cavity. As an alternative, micro welding is being used.

Method 2 uses inductive coupling (IDC) between the card antenna and another small antenna which is part of the contact chip module itself. This method requires a special chip module with an integrated coil and a specially shaped booster antenna in the card body. On the other hand, the embedding of the chip module to the card body is no different to a contact-only card.

2.2.2.6 Preparation of the Chip Module

The chip module of a smart card consists of the chip embedded into a chip module and offers the six or eight contact plates (pins) to the card Reader terminal or mobile device.

The chip hardware platform is produced in semiconductor factories. The silicon wafer fabrication plant (wafer fab) creates the electronic structures on the silicon. After sawing, the individual dies are placed on carrier tapes with lead frame. The front side of the carrier tape represents the metallised contact plates of the future smart card. By wire bonding, the chip and reverse side of the contact pads are connected. After encapsulation with the mould, the electrical and visual inspection finishes the process. As alternative to wire bonding, flip chip technology can be applied. Instead of wires, contact pads are used to connect chip and contact plates. For this, the orientation of the die within the module is rotated by 180° flipped face down.

In most cases, chip modules are shipped on reels to the smart card manufacturer. In a first step, an incoming inspection will be made on a test handling machine to ensure the quality of the modules before embedding. Usually the Answer To Reset (ATR) of the chip is checked and read/write tests on the Electrically Erasable Programmable Read-Only Memory (EEPROM) are performed.

As machine costs for test handlers are lower than for card personalisation machines, they are often used to load data to the chip which are common for a range of products.

For ROM masks this is the completion of the ROM OS in the EEPROM, e.g. for extensions and patches. For Flash controllers the complete OS has to be loaded in this step.

Depending on the contents of the initialisation file loaded afterwards, file structures and also partly their contents will be created, applications and keys will be available on the chip.

The criteria which parts to load in which step will not only be dependant on cost calculations but also on the product and related security requirements. So for some products, it is necessary to have a clear separation between the initialisation and the personalisation. For other products, it may be better to perform the initialisation as late as possible. This avoids logistic problems, as not too many variants have to be kept in stock for the subsequent processes.

As the modules have to be glued into the card body, another step is necessary before embedding which applies an adhesive tape to the modules.

2.2.2.7 Milling, Implanting and Punching

Before the smart card module can be embedded, a cavity needs to be milled into the card body (of course this step does not apply for injection moulded cards which already have it).

Now the "marriage" of card body and the module can happen. The module will be applied and glued to the card body in an implanting machine. To verify that the module is still alive after this process step, usually an ATR test is performed in the implanting machine. For some products, it is also necessary to write some information onto the card body. This is possible via an inkjet printer within the machine.

For mobile Phone cards (Subscriber Identity Module—SIM) finally a punching of the Plug-In is needed. Depending on the type of the punch (Form Factor) different tools are used in the machine.

2.2.3 Personalisation and Related Services

In personalisation, an individual product will be created for the end customer. The data necessary to do this is usually provided by the card issuer—sometimes enhanced by data generated in the process at personalisation (Fig. 2.4).

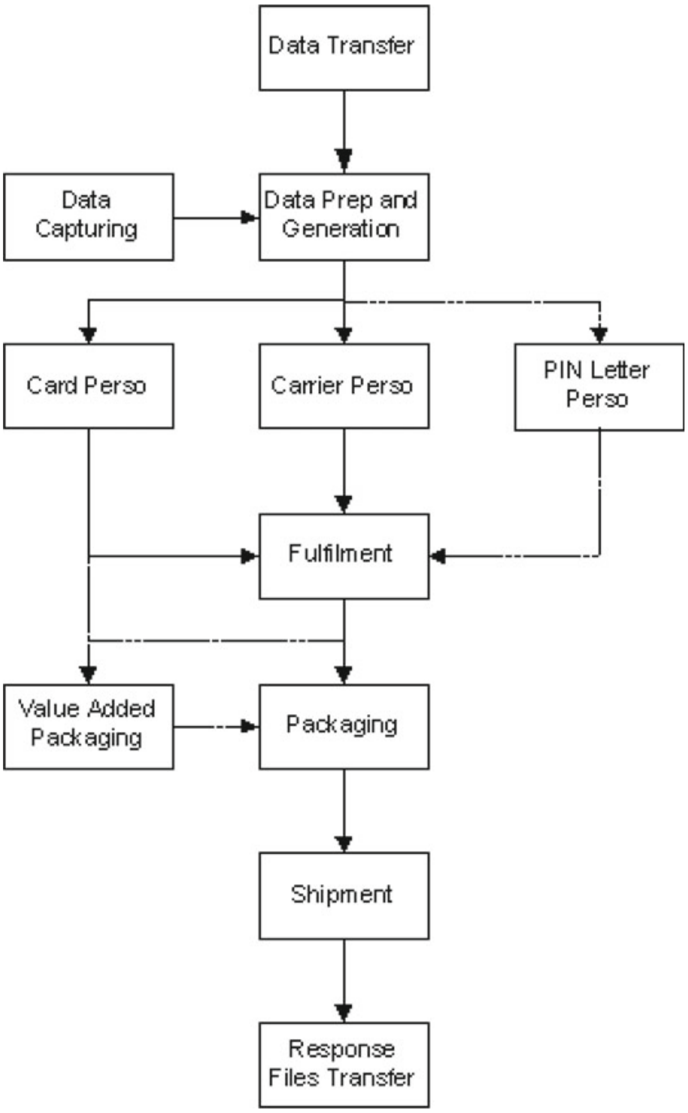


Fig. 2.4 Personalisation and related services flowchart

2.2.3.1 Data Transfer

Customer data usually enters the smart card via Integrated Services Digital Network (ISDN) dial-up data connections or encrypted channels over the Internet.

The data has to be encrypted and will be decrypted only after being transferred to the production network. Depending on the card issuer's system and the number of products, several files will be provided. Quite often similar products are within one file, e.g. Visa and MasterCard credit cards.

2.2.3.2 Data Capturing

In most cases, the card issuer will supply the data of his customer, but there are also data capturing services performed by personalisation bureaus. In a typical scenario, the customer application form for a card has a special part for the photo, which is torn off and sent directly to the personalisation bureau. The photo will be scanned and stored under a reference number, so it can be linked with the other personalisation data sent by the card issuer to create a photo card.

2.2.3.3 Data Preparation and Generation

Due to security requirements, the production network is logically separated from the data transfer network. So before the data can be processed, a transfer of the data via the separating firewall has to be initiated.

After decryption of the files a validation of the data takes place. Sometimes also a conversion has to be done, e.g. for banking card issuers with mainframe systems it is quite common to convert their Extended Binary Coded Decimal Interchange (EBCDIC) data to American Standard Code for Information Exchange (ASCII) before further processing.

For validation first the file structure and integrity will be checked, and then also whether the data fields contain allowed values. This may be simple checks like whether a field is numeric or checks whether there is a defined product and process available as requested by control fields of the customer data.

In many cases, a grouping and sorting of the data will be the next task. So there may be different service levels and certain records have to be processed and produced on the same day while for others there is a bigger timeframe. Other criteria may be different shipment methods (by mail, by courier, etc.), different enclosures to go with the card mailing or different addresses of the card issuer's locations where the cards should be sent to.

A merge of data from different sources is another task of data processing. This may be photos or logos for optical personalisation as well as data for different applications on the chip.

For many products, it is also necessary to generate additional data which will go into the chip. This is very common for SIMs, where the network operator often only provides the basic numbers (Integrated Circuit Card Identifier—ICCID and International mobile Subscriber Identity—IMSI). The values for keys (e.g. the Ki) and secret values (e.g. PIN, Personal Unblocking Key—PUK) have to be generated with a random generator or are derived by using certain card issuer keys and/or calculation methods. In that case the card issuer needs to receive a response file which contains all the values generated, so he can store it in his systems. Another task for products being sent out by direct mail is to create the postage information—depending on mail type, weight and destination. Due to the requirements of the local mail service this information needs to be printed on the carrier, probably leads to the usage of different envelopes in fulfilment and must be provided in a billing report.

Also for credit cards with chip (EMV), there is a process which takes the magnetic stripe data and some card issuer keys to generate data for the chip.

As the secure storage and generation of keys and the encryption of production data is a basic requirement today, a key management system and the usage of Hardware Security Modules (HSMs) is a must.

At the end of the data preparation process there are several outputs: data validation reports, production files for the different card personalisation machines, printing files for carriers and labels, information on the products (bill of material, process steps) for production and sales (what to bill to the card issuer?), log files and audit trails and also response files for the card issuer.

2.2.3.4 Card Personalisation

Depending on the product, machines have to offer a wide range of personalisation technologies. Most machines can be set up individually by combining different machine modules and can also be adjusted for further requirements later (Fig. 2.5).

Laser Engraving

Laser engraving is the most secure way of optical personalisation. Its result can be seen in the different layers of a card and be felt on the surface of the card. The laser can either personalise vector fonts or raster images. The latter one takes more time, so for bigger images (photos, logos, barcodes) it is necessary to have more laser modules in one machine for a high output.

Another advantage for laser personalisation is reduced cost, as no ink or transfer film is needed.

Embossing and Indent Printing

Embossing and Indent printing are the classical methods for personalising credit cards. Still in many countries credit cards are not processed online, so the embossed characters are needed to create the receipt. Embossing is done with typewriter wheels with standardised font types (OCR-A, OCR-B). In modern high speed machines two or more modules are used to enable high throughputs. Printing on the rear side of the card is called indent printing, characters are not embossed in that case.

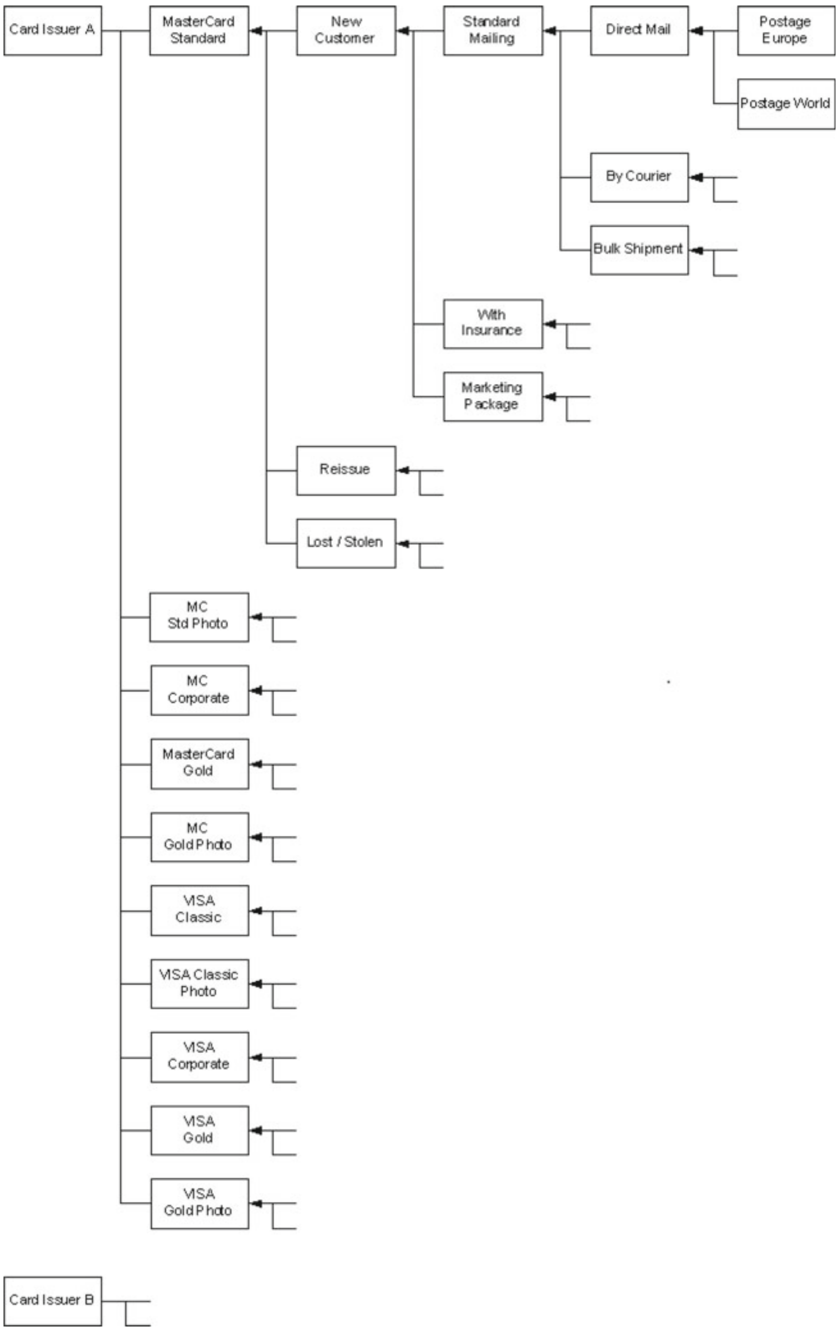


Fig. 2.5 Variants in personalisation (Example)

Inkjet

Inkjet printing is often used in conjunction with simple products such as voucher cards. There are machines available which have a very high throughput (40000 cards per hour). On the other hand, inkjet also can be used for colour images.

Thermal Transfer, Colour Dye Sublimation and Retransfer Printing

These techniques work with ink ribbons and thermal print heads. Thermal transfer printing is used for monochrome images, such as logos or barcodes. It delivers high optical quality, but less security than laser engraving, as the ink is applied to the surface only and does not go into the deeper layers.

For colour dye sublimation a three-pass process is necessary, using ribbons for Yellow, Magenta and Cyan. Usually an overlay ribbon is applied on top of the images to protect them against abrasion and fading. Again, the images are only on the surface and therefore not as secure against copying as laser images.

The retransfer method is similar to colour dye sublimation, but instead of printing directly to the card a reverse image is printed to a transfer film which is then applied to the card body. The main advantages are better quality, as an unevenness of the card does not affect the printing result and that the image can be printed over the full surface area of the card and no white borders can be seen. This technique is used for "picture cards" where the end customer may choose his personal design from a given choice of pictures or even by sending his own photo taken with a digital camera.

Magnetic Stripe Encoding

To encode the magnetic stripe with its three tracks (e.g. for credit cards track 1 and 2 are used) magnetic stripe readers are used. Usually there's at least two of them in a machine: the first one encodes the magnetic stripe, the second one reads back the information from the magnetic stripe to ensure that it is written correctly. It is also possible to pre-encode a card's magnetic stripe to use that information in personalisation—e.g. to check whether the right plastic is used.

Chip Encoding

Chip personalisation has become quite a complex process in the last few years, as the capabilities of the chips (e.g. JavaCard), the memory sizes (Megabytes!) and also the security requirements have increased.

The basic process is that the card reader has to establish a connection to the smart card, perform an authentication by presenting a key and then select files on the smart card and update them with personalised contents provided by the data preparation and generation process. Depending on the complexity of the product, these steps may involve a mutual authentication between card and reader, the use of an external HSM to handle/create keys and encrypt/decrypt the communication channel. Additional data may be loaded from different sources (configuration files, databases) or also be generated during the personalisation process and passed back to data preparation. So the smart card itself may perform asymmetric key generation and export the public part for a certificate request.

To cope with the amount of data and the throughput needed, a number of high performance card readers are needed in personalisation machines. It must be possible to change parameters like voltage, frequency or divider in a wide range for optimisation. With local memory available on the readers, also certain parts of the personalisation data can be stored there to improve performance. Current products also offer new smart card protocols, like USB or SWP (Single Wire Protocol). There are also high requirements to the hard- and software handling the personalisation data and process—regarding quality, performance and stability. A typical scenario at the time of writing is to handle 60 smart cards in parallel and load each one individually with some hundred kilobytes with other components involved (HSMs, databases etc.).

One of the challenges of the smart card industry is to cope with the increase in memory sizes now available in chip modules. While for years there were only a few kilobytes to handle in the initialisation and personalisation, with the propagation of flash technology today a typical SIM card will be loaded with a few 100 kilobytes. And SIM products with 512 megabytes or more additional memory are available.

As the encoding times are limited by the standard protocols (T = 0, T = 1) and in the end physically by the memory write times, personalisation machines have to be equipped with a number of card readers to achieve a reasonable throughput. Let us illustrate that with an example: for a throughput of 3,600 cards per hour the machine cycle time is only one second for each card. If the chip encoding time is 40 s, we need 40 card readers working in the machine in parallel to keep this throughput.

Typical Personalisation Machines

As mentioned before most personalisation machine vendors offer a modular design of their machines to fit a wide range of requirements. The input and output modules either handle a loose stack of cards or work with magazines. Some machines may have more than one input module, so different plastics can be mixed in personalisation. If the same plastic has to be separated for different card issuers or sorted for later shipment more output stacks are an option as well.

A typical machine for credit card personalisation will have a magnetic stripe Reader module, followed by a chip encoding module for EMV cards. With a thermo transfer module (for front and/or rear side) an additional logo can be placed on the design—e.g. a company's logo for company credit cards. A colour dye sublimation module may follow to personalise a photo of the cardholder—again this may be for the front or rear side. The last stations in the machine will be the embossing units, one with types to emboss the credit card number, one or more (for high throughput) other units to emboss the remaining lines, e.g. the cardholder name. The performance range starts with 200 cards per hour (cph) for small desktop systems and ends at 3000 cph for high volume systems.

A typical machine for SIM card personalisation may have a vision system after the input module, which serves two purposes: verify that the right card body is used and calculate offsets for the origin for optical personalisation to equalise punching tolerances. As the data volume for the chip can be quite high for SIM cards, there will be multiple chip encoding heads working parallel to ensure the machine throughput does not go down for longer loading times. High volume machines which run at more than 3000 card-per-hour may have 40, 60 or even more chip encoding heads. Many SIM cards will only receive an optical personalisation with a number (the ICCID), some may also have a barcode. So the typical number of laser stations is one or two. To be flexible for either front or rear personalisation often a flip over station is used. To verify the quality of the optical personalisation, a vision system can be the last module before the cards go to the output stacker.

If cards can not be processed (e.g. if the chip does not work) the machines will treat them as rejects and put them on a separate reject output stack.

Additional modules are available for printing the card carrier, affixing the card to the carrier and also to put this in an envelope, probably together with some additional enclosures. Higher volumes are often handled on separate machines instead of this "inline process". This will be described in the next two paragraphs.

2.2.3.5 Carrier and PIN Letter Personalisation

The letter to the end user which carries the card is in most cases personalised using a laser printer. For small volumes this may be simple office printers, for high volumes there are high speed machines printing up to 250 pages per minute (ppm) for cutsheet printers or even over 1000 ppm for continuous feed printers.

Most card issuers today provide a blank paper which only has their pre-printed logo and probably some fixed text on the rear side. All the rest will be printed variable, which gives the card issuer a maximum of flexibility and enables them to address their customers very personally. For smaller volumes colour printers are an option as well, which will print all information including logos etc. on a white sheet of paper.

To enable an automatic matching of the card and the carrier in the fulfilment step, a machine readable card identification number needs to be printed onto the carrier—either as barcode or using an OCR (Optical Character Recognition) font type.

PIN (Personal Identification Number) letter personalisation today most often also works with laser printers. One method is to cover the PIN with a sealed label after printing, another one works with a special paper which already incorporates a sealed label.

Another method still used works with needle printers and carbon coated multilayer paper. There is no carbon ribbon in the printer, so the PIN cannot be seen during printing—but will be found in the PIN letter after tearing off the seals.

2.2.3.6 Fulfilment

The most common way to hand out a card product to the end customer is to attach it to a personalised letter (carrier), optionally add some information leaflets (enclosures) and put it all into an envelope. Depending on the card issuer requirements this leads to a high variety of products, e.g. by sending out the same card product with different enclosures.

The process may either be handled manually for small batch sizes or by dedicated mail processing systems for higher volumes, with a throughput of up to 8,000 mailings per hour. These machines are set up from different modules; a typical configuration looks like as follows:

The paper feed module will take the pre-printed carriers, in case of continuous paper a cutter will then cut it into single sheets. In the next module an adhesive label will be placed onto the carrier. The card attaching station reads information from the card (usually from the magnetic stripe or chip) and the corresponding information from the carrier (usually OCR or barcode). If the information matches, the card will be affixed to the carrier. It is also possible to attach more than one card to the carrier—so a bank may send out a MasterCard and a Visa card on the same carrier to its customers or a family receives all their health cards within one mailing.

Afterwards the carrier with the attached card is folded (e.g. z-fold, wrap fold) and inserted into the envelope. An inserter module consists of a number enclosure stations from which for each mailing additional enclosures can be individually pulled and inserted. Most often these are non-personalised information leaflets or booklets, but also personalised items are possible, e.g. by matching them via a barcode.

After all components are in the envelope, it is sealed and a weighing scale behind it checks the weight of each mailing. This may be used to check whether the mailing is correct (e.g. the card did not get lost in the machine) and to calculate postage or to sort out different mailing types as well. From the output stacker module the operator can the take the mailings and put them into boxes, which will then be handed over to the shipment area.

2.2.3.7 Packaging

For products which are not sent directly to the end customer, packages have to be made according to the logistical needs of the card issuer. These packages may either contain only cards or cards in mailings which are distributed in other ways after they leave the personalisation bureau. Related to packaging there is the printing of shipment lists and identification labels to the cardboard boxes. As these lists and labels apart from some overall product information also contain some personalisation data (e.g. first and last card number in a box, on a pallet) the data has to be provided by the personalisation data processing systems.

2.2.3.8 Value-Added Packaging (VAP)

To address customers even more individually, there are many variants to packaging of cards. In nearly all cases this is a manual process, as the individual items cannot be handled by a machine and the volumes usually are not high enough for the investment in an automated solution.

Typical products for VAP are gift or SIM cards. There’s a wide range of boxing available, e.g. CD boxes, sophisticated cardboard boxes, blister packages and even wooden boxes or leather cases. With the cards may go user guides, mobile phone handsets and manuals or different marketing items.

2.2.3.9 Response Files

Last but not least, there are also non-physical products which have their origin in personalisation. During data generation and processing some data is created which the card issuer may need in his systems for either logistical or technical reasons. Some card issuers need the information when which card number has left the personalisation site. On the one hand, so they are able to answer requests from their customers ("When will I receive my card?"); on the other hand, this information may be necessary for them to start a related process, e.g. printing and sending out a PIN letter.

Whenever a card contains individual values created or allocated in personalisation, the card issuer will need these values in his system. This may be individual card keys like the Ki for GSM cards which is needed in the provider's authentication system or a chip hardware identification number to be stored in a CAMS (Card Application Management System) for later purpose. For some processes it is even necessary to send a response file to the card issuer or another related party and wait for an answer to this response file to continue production. For example this applies to load certificates for keys generated on the smart card: during personalisation the smart card generates an asymmetric key pair, the public key is sent in a certification request to an external certification authority (a trust centre) and the certificate received gets personalised to the smart card in a second personalisation step.

2.2.3.10 Logistics

One of the challenges in personalisation is to handle logistics most effectively. Due to the many variants which are generated by different card bodies, carrier papers, enclosures and shipment methods production breaks down into small lot sizes. On the other hand, there are very restrictive rules how cards have to be treated in a secure production environment. The cards are stored in a vault and any movement and withdrawal has to be recorded. A counting of cards takes place between the significant process steps. When a card is spoiled in a process, this of course has to be recorded as well. At any time a "four-eyes-principle" has to ensure the integrity of the process.

There are two main ways to provide the cards to personalisation: either the amount of cards given by the card issuer order is moved to personalisation and rejects produced on the machines have to be pulled in an additional run—or a higher amount is moved to production and the rest needs to be balanced at the end when returned to the vault. In order to reduce machine setup times, similar orders can be processed together—this can also be supported by intelligent data preparation. For example, if there are four different card designs which are applied to the same type of carrier paper, the data preparation will create one carrier printing file and four card embossing files. So there is only one order at the printer instead of four. The same then applies to fulfilment where the card stacks are combined and the machine can produce with one carrier stack in one run.

2.2.3.11 More Individual Products and Services

In the last years there has been an emerging trend to supply even more individual products and services to the card issuer. It starts with a high variety of card body designs. So a typical bank portfolio comprises MasterCard and VISA credit cards—standard/gold, private/business, in cooperation with other companies (Co-Branding), special editions for an event (e.g. World Cup, Olympics), with and without photo, etc. On top of that there are debit cards, customer cards, savings account cards—again with different characteristics. Up to fifty different designs per card issuer is not unusual, some even have hundreds.

This leads to small batches in card body production and to even smaller lot sizes in personalisation. With the "picture card" this reduces to lot size of one: the individual card design is made from a digital photo provided by the cardholder.

Variants which can be handled quite easily in production are different texts which are printed on the card carrier. Many card issuers have dozens of text variants to send out the same card on the same carrier to address their customers individually. The print programmes will print the different variants in one run, but there is considerable effort to create and maintain the related templates. If the carrier papers also shall be in many different designs, colour printing on demand is a possible solution.

The next level of variety comes with packaging. Much of it can be handled with mail processing systems, which are able to pick different enclosures to create the mailing. But for special formats, enclosures that are not capable of machine handling and VAP, manual work is often necessary.

Finally there are different shipment methods which expand the number of variants. There is bulk shipment using different carriers, there is direct mail with the established post or alternative service providers and there is shipment by different couriers. A related service requested from card personalisation bureaus is to pull certain cards during production and switch their shipment type, e.g. from direct mail to courier.

The card issuer today expects narrow service levels. For the introduction of new card body designs this is still weeks, smaller changes of a product configuration may be already requested on a day-to-day basis. Card personalisation and shipment for many products (in the daily low-volume business) are handled within 1 or 2 days, but for some products the timeframe is even limited to hours (e.g. emergency credit cards).

2.2.3.12 Data Management

The main business for personalisation bureaus today still is "data in—card out—delete data". But more and more card issuers also ask for management of their data and request more detailed information on their orders during the production process.

Managing data starts with such simple solutions like the storage of scanned photos or a secure storage of the keysets on a card, which may be recalled for later customer applications. Another more complex example is an Internet gateway for the cardholder which enables him to select an individual card design or to provide his digital photo for his personal credit card.

When it comes to completely manage the lifecycle of a card, enable post issuance personalisation (e.g. adding a new application to a smart card after the cardholder already received his card) and store and manage all related information, a Card Application Management Systems (CAMS) is required.

As personalisation bureaus handle material on behalf of the card issuers (cards, carrier paper, leaflets, envelopes etc.), the card issuer either needs continual information to control his stock levels or can ask his supplier to do so for him. Monthly stock reports often are still state of the art, but card issuers more and more ask for online and actual access to this data.

Similar requirements are known for a more detailed and actual view on the production process. Card issuers want to view the progress of their order—even down to the cardholder level. As on the other hand requirements on data security are very high, this is not easy to implement.

2.2.4 Security and Quality

Security is one of the main issues in smart card production. A card manufacturing plant or personalisation bureau has to fulfil high requirements on physical security. This starts with the fences around the building, which must constructed in a way that no car or lorry simply can break through it. Additional electronic systems detect any other trials to break through this first barrier. Video cameras need to survey the whole plant area as well. The building itself has to fulfil certain standards (wall thickness, stability of doors, etc.) and of course especially in the production area.

The security areas may only be entered via mantraps and there are clear policies for any access necessary by non-registered staff (e.g. for service of production machines). Only people who are able to prove their integrity may work in those areas and all their comings and goings are recorded. No single person is allowed in the security area, a four-eyes-principle needs to be guaranteed in any case, supported by video cameras all over.

Security is also part of all processes—there is a continuous counting of security relevant materials, such as cards, holograms etc. during an order workflow. Another very important task to ensure logical security in personalisation is the protection of data. Networks for smart card production are strictly separated from other networks and of course from the internet. Access to data is limited to the persons who need to deal with it and encryption of data is applied wherever possible. It is also essential to delete the personalisation data after production in a safe way. On the other hand certain data has to be kept on behalf of the card issuer or to ensure traceability.

Organisations such as MasterCard, VISA or the GSM Association will perform regular audits to prove the physical and logical security in card production sites. If severe problems were detected by them, this could lead to a decertification and such to the loss of the business. Therefore an ongoing process has to be established, which always ensures the compliance with the actual security regulations.

While the card industry formerly was mainly concentrated on the physical security of the production sites—including walls, doors, vaults, mantraps, video systems etc.—today the focus is more on the logical security.

As personalisation bureaus handle such sensitive data as credit card numbers, their processes and related IT systems have to be on a very high security level. This involves the secure handling and storage of keys and data, a reliable firewall concept, an effective data access restriction and a gapless monitoring of all activities in the network—right up to the operation of Intrusion Detection Systems (IDS).

The encryption of cardholder data throughout the whole process is one basic requirement, even though the data is already kept in separated network and most of the data can be seen on the card and the carrier during the production process.

Even higher security levels can be achieved by the separation of different firewalled segments within a production network. This may also be necessary to segregate data from different card issuers.

The access to data has to be based on a "need-to-know" basis. The requirements here often exceed the capabilities of the on-board functionality of operating systems. So the applications need to establish an additional layer, e.g. by implementing four-eyes-principle, the use of smart cards for access etc.

There is considerable effort to implement such a consistent data security and as it also leads to more complex processes, there are additional ongoing costs as well.

Quality is the other very important issue. Well defined quality management procedures and a quality assurance during the whole product lifecycle are a matter of course. This starts with the definition of a product, continues with development, test and the production release process following it. During the production various sample tests and in some cases 100% tests are performed to finally prove the quality of the products before they get shipped. Examples are:

- Visual and electronical control is performed after printing processes
- An incoming inspection on modules checks the physical dimensions and the EEPROM as well
- A depth control is performed on the milled cavity before embedding the module
- Cards are tested in bending and torsion cycles to ensure they and the smart card module are fit for use
- Personalised cards are checked against the personalisation data to ensure that the right data got onto the card
- Sample mailings will be opened to ensure that the end customer gets the product in the right configuration
- Response data is checked against personalised cards to ensure it matches to the product delivered to the end customer

Everyone can imagine that all these investments in security and quality lead to very high initial and ongoing financial efforts. This means high hurdles for newcomers in the market and a challenge for the existing companies to remain competitive.

2.2.5 Current Trends

2.2.5.1 Industrialisation and Commoditisation

The volume of smart cards with microprocessor chip shipped to the global markets grew from approx. 5.5 Bn units in 2010 to over 9 Bn units in 2015. This rising demand in combination with increasing cost pressure in the two main areas of smart card deployment is calling for industrialised solutions in smart card manufacturing. The typical industrial challenges of meeting cost targets and short lead times in combination with zero defect quality levels fully apply.

Industrial principles and state-of-the-art production systems are applicable both to the manufacturing of the smart card and the fulfilment of personalisation services.

The design of global supply chains has to meet the dedicated requirements of global and local customers especially in the areas of Financial Institutions and Telecom Industries. A smart card can never be produced as a customer-neutral stock item. The variant creation typically starts early in the process—printing of issuer-specific artwork is one of the reasons. Later, the loading of the smart card operating system and its application profiles further customises the product, for mailing services, individualised welcome letters are generated. Despatch routes often include personal courier services to individual card holder level. Flexible solutions for meeting these requirements is key for the global smart card industry. There are mature markets such as in Europe but also still emerging areas where the migration from non-card or at least non-smart card applications to smart cards will keep momentum up.

2.2.5.2 Example for Complementary Technology—Subscription Management

The main purpose of smart cards in the form of a SIM card was initially securing the personal mobile communication. As an additional use case, SIM cards are now used for the secure connectivity of devices. Machine-to-machine communication (M2M) is required for applications in connected cars, connected energy and connected logistics. The subscription credentials of SIM cards in any of the known form factors are required just as for the traditional use case in mobile phones.

Subscription management is the technology for responding to a requirement that arose from the fact that embedded secure elements (SIM cards) may have to be re-configured during the lifetime. Loading of subscription credentials over-the-air (OTA) allows application of such changes without replacement of the SIM hardware itself. The advantages of a hardware-based secure element in form of a SIM card are kept but the efforts for maintenance of devices in the field is substantially reduced.

For this purpose end-to-end management of the entire SIM and device lifecycle is necessary, from activation and subscription credential management to deactivation. Subscription data processing, secure key handling and distribution of messages to M2M devices requires secure and certified data centres.

The generation of initial subscriptions is still connected to the data generation and personalisation process of the SIM. The provisioning credentials are required for activation and deployment of subsequent steps after delivery of the SIM to the end user or to the system integrator in case of an M2M application.

2.2.5.3 Example of Disrupting Technology—Host Card Emulation

Host Card Emulation (HCE) is a technology which replaces the hardware of a secure element by a fully software-based solution which can be executed in standard environments without special hardware security measures. Thus, smart cards would become obsolete. In addition, all hardware-related components including physical validation would be completely lost.

The trend towards HCE may be followed by some areas and for distinct applications. A transition from physical to purely virtual security is an option for the connected infrastructures. However, the smart card as secure document will still keep its justification and as a consequence, the enhancement of the physical characteristics of smart cards and their advanced manufacturing technologies will keep its relevance for the future.

2.3 Summary

Smart card production entails a broad selection of activities and technologies. It begins with printing, laminating or injection moulding of the card body and the application of several card related items such as magnetic stripes and holograms. It is followed by the test and initialisation of the smart card modules which are then embedded into the card body.

In the personalisation process of the card there is a physical part performed by laser, embossing and thermo transfer modules, and an electrical part whereby the magnetic stripe and the smart card module (with contacts or contactless) are encoded. Personalisation also happens for card carriers, followed by related services such as mail processing, packaging and shipment. All these steps are strongly linked with a data preparation process.

One of the current trends and challenges is the demand for even more individualised products and services, including an extended data management process. Others trends are the increasing memory size of smart cards with its related impact on personalisation times and the increasing security requirements, which need to be supported by appropriately improved IT architectures.

Quality and security are the most important aspects in smart card production and mature audited processes need to be implemented at all times.

The still growing global demand for smart cards requires manufacturing and personalisation of smart cards as fully industrialised process and embedded into a global supply chain. Subscription Management as well as Host Card Emulation may become game changers for the volume markets for smart cards—Telecom Industries and Financial Institutions.

"This article is the result of the experience of the authors and colleagues at G&D, who we would like to thank for their kind support. It represents the authors' personal views only and not necessarily those of G&D or any of its affiliates."

Useful Websites

The reader may find the following websites useful:

- http://www.icma.com
- Site of the International Card Manufacturers Association

- http://www.gi-de.com
- Site of Giesecke & Devrient

Glossary

3FF	3rd Form Factor
4FF	4th Form Factor
ABS	Acryl Butadiene Styrene
ASCII	American Standard Code for Information Interchange
ATR	Answer To Request
CAMS	Card Application Management System
Cph	Card per hour
CTP	Computer to plate
DI	Dual Interface
EBCDIC	Extended Binary Coded Decimals Interchange Code
EEPROM	Electrically Erasable Programmable Read-Only Memory
EMV	Europay, MasterCard, Visa
FF	Form Factor
GSM	Global System for Mobile Communications
HCE	Host Card Emulation
HSM	Hardware Security Module
ICCID	Integrated CirCuit Identification
IDC	Inductive Coupling
IDS	Intrusion Detection System
IMSI	International Mobile Subscriber Identification
ISDN	Integrated Services Digital Network
Ki	Individual Subscriber Authentication Key
LTE	Long Term Evolution
M2M	Machine-to-machine communication
MFF	Machine Form Factor
MNO	Mobile Network Operator
OCR	Optical Character Recognition
OS	Operating System
OTA	Over-the-air

PIN	Personal Identification Number
PC	Poly Carbonate
PCB	Printed Circuit Board
ppm	Pages per minute
PVC	Poly Vinyl Chloride
PET	Poly Ethylene Terephthalate
PETG	PET Glycol-modified
PIN	Personal Identification Number
PLA	Poly Lactic Acid
PUK	PIN Unblocking Key
ROM	Read Only Memory
S@T	SIMalliance Toolbox
SE	Secure Element
SIM	Subscriber Identity Module
SM	Subscription Management
SMD	Surface Mount Device
SWP	Single Wire Protocol
UICC	UMTS Integrated Circuit Card
USB	Universal Serial Bus
VAP	Value Added Packaging
WiB	Wireless Internet Browser

References

1. Yahya Haghiri, Thomas Tarantino: *Smart Card Manufacturing: A practical guide*, John Wiley & Sons Ltd, 2002.
2. ISO/IEC 7810. More Information Available via http://www.iso.org/iso/catalogue_detail?csnumber=31432, accessed August 2016.
3. ETSI TS 102 221 Smart Cards; UICC-Terminal interface; Physical and logical characteristics. More information at http://www.etsi.org/deliver/etsi_ts/102200_102299/102221/08.02.00_60/ts_102221v080200p.pdf, accessed August 2016.

Chapter 3
Multi-Application Smart Card Platforms and Operating Systems

Konstantinos Markantonakis and Raja Naeem Akram

Abstract Although smart card technology has been available for many decades, it is only in the last few years that smart cards have become widely considered as one of the most common secure computing devices. They are encountered in a number of applications (e.g. secure wireless access in mobile networks, banking, identification) satisfying a diverse range of uses. One of the fundamental factors contributing towards the success of smart card technology is tamper resistance. As the underlying smart card processing power increases at a constant pace, more and more functionality becomes available. It was soon realised that in order to grasp the full benefits of the underlying hardware, parallel advances in the corresponding smart card operating systems would be necessary. This chapter provides an overview of the most widely utilised smart card operating systems or platforms that enable multiple applications to be securely managed and reside in the same smart card.

Keywords Java card · MULTOS · GlobalPlatform · Smart Card Operating System (SCOS) · Multi-application Smart Cards · WfSC · .NET · BasicCard

3.1 Introduction

Smart cards are already playing a very important role in the area of information technology. Smart card microprocessors are encountered as authentication tokens in mobile phones, in bank cards, in tickets in the transport industry, in passports and identification cards. Since their invention [1], during the 1960s and 1970s, they were considered as a portable medium for secure data storage. Further key features

K. Markantonakis (✉) · R.N. Akram
Smart Card Centre, Information Security Group,
Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK
e-mail: k.markantonakis@rhul.ac.uk

R.N. Akram
e-mail: r.n.akram@rhul.ac.uk

K. Mayes and K. Markantonakis (eds.), *Smart Cards, Tokens, Security and Applications*, DOI 10.1007/978-3-319-50500-8_3

of smart card technology that can be considered to contribute towards their wider deployment are security, programmability and upgradeability [2].

It is obvious that the smart card hardware and software are very closely coupled together and they both contribute, along with other factors, towards the success of smart card technology. It is only in the late 1990s that the concept of issuing smart cards and adding code later on (or further functionality), even when the card is in the cardholder's hands, came into existence. This concept is known as post-issuance capability.

In order to offer such functionality, it is essential that both the hardware of the underlying smart card microprocessor and the software are sophisticated enough to address all the additional requirements. In this chapter we will examine several of these platforms (mainly multi-application), classify them and at the same time identify their advantages and disadvantages.

3.1.1 Smart Card Platform Evolution

Early in the 1990s, the available smart card technology was offering 1–3 kbytes of ROM (Read Only Memory), less than 128 bytes of RAM (Random Access Memory), and approximately 1–2 kbytes of EEPROM (Electrically Erasable Programmable Read Only Memory). The underlying software platform comprised a number of program routines, often masked in ROM, that allowed the outside world to communicate with the card in a controlled manner. The way in which an entity could communicate with a smart card was specified by the ISO 7816-4 [3] series of standards. This included amongst other things support for simple file management operations (e.g. write/read a block of data) and certain cryptographic operations.

The main reason behind placing most of the smart card functionality in the ROM of the card was that ROM was (and still is) using very little chip area per bit [4] within the overall size of the smart card microprocessor. Additionally, smart cards were mainly utilised for a dedicated function, i.e. they were single application oriented. Economies of scale were also an important influencing factor. If the number of cards was large then it was justifiable for smart card manufacturers to invest in the development of a dedicated ROM mask that offered the necessary functionality. The structures of these cards were often referred to as monolithic as it was impossible to change them after they were embedded in the card. The smart card manufacturers provided certain solutions around this problem [4], mainly by allowing transfer of execution from ROM to certain jump tables within the EEPROM containing the necessary code improvements. However, programming ROM involves a long and expensive development cycle that requires a lot of skills. Thus, the need to be able to quickly develop and securely execute a smart card application became evident.

The next generation of smart cards that claimed to bypass some of the aforementioned problems, introduced the concept of a smart card operating system. A smart card operating system is often defined as "the software that manages the sharing of the resources of a computer" or "a software platform on top of which other programs, called *application programs*, can run. The application programs must be written to run on top of a particular operating system" [5].

The development of smart card operating systems followed a similar path to the development of operating systems in traditional computing devices. The main aim was to offer a stable platform that would allow smart card application execution without suffering from the aforementioned problems. Although during the mid-1990s, a number of companies claimed that they offered smart cards with powerful smart card operating systems, the reality was different. As mentioned above, the smart cards were mostly considered as secure storage devices. Furthermore, the smart card application developers had very limited flexibility as the smart card applications were still mainly developed in ROM and in pre-agreed application structures. Smart card applications had to be developed for specific smart card microprocessors. Additionally, most of the card functionality was embedded with the smart card operating system (SCOS). A direct consequence of "security through obscurity" (under which smart card manufacturers attempted to keep confidential any information regarding their products) was that there were very few smart card programmers. Moreover, it was almost impossible to find information on how to program cards and more importantly to obtain tools that would allow you to develop smart card applications.

Portability was an issue as a smart card application developed for a smart card utilising a specific smart card microprocessor could not run on a different smart card with a different microprocessor. Platform independence was a real problem that required applications to be rewritten from scratch in order to become portable within different platforms. However, around this period a number of smart card operating systems came into existence, e.g. MPCOS from Gemplus, STARCOS from Giesecke & Devrient (G&D), CardOS from Siemens, OSCAR from GIS. Still these smart card applications were very closely coupled together with the underlying smart card operating system and often everything was stored in ROM.

The requirements for adequate memory management, support for state machines, portability and interoperability and advances in microprocessor technology led to new initiatives. The main goals were the complete independence from the underlying operating system, the ability to securely handle multiple applications and also being able to securely modify the content of the card after it was issued. It is evident that security was already playing a very important role in smart card technology, but it was brought further forward in an attempt to provide the necessary reassurance that more openness would not introduce any vulnerability.

This was further realised between 1995 and 1999 with the introduction of more powerful microprocessors offering approximately 6–8 kbytes of ROM, 128 bytes of RAM and 5–12 kbytes of EEPROM. It is very easy for anyone to realise that the above hardware platform does not offer great flexibility for the design of an operating system within the traditional meaning of the word. However, someone might claim that due to the limited underlying hardware resources, the design of a SCOS should not be a difficult task. Evidently, the underlying hardware has considerable influence in the design of smart card operating systems. Improving the overall performance of the smart card microprocessors was among the main topics of discussion within the smart card communities. In particular, the performance of cryptographic operations was drawing a lot of attention from the academic community. It was however misunderstood that cryptography was among the main delaying factors within a smart card. The communication buffer (usually between 128–250 bytes) that allowed the card to communicate with the outside world, was often overlooked [7].

Its was approximately around this time (last part of the 1990s to early 2000) that the improvements in smart card hardware made it possible for further ideas to evolve and become more realistic. For example, the idea of extending the SCOS functionality, but using patch/extension tables is already mentioned. However, what was needed was the ability to offer a secure and constrained environment that would provide guarantees for controlled application execution. The concept was not new, especially in the traditional computing environments. This was exactly what was offered by a number of source code interpreters (e.g. Microsoft Basic [8]) in early home computers.

The main efforts for the provision of multi-application smart card platforms took place between 1997 and 2001, mainly through the introduction of the following distinct smart card platforms, namely Java Card [9–11], MULTOS [12, 13], GlobalPlatform [2, 14, 15], Windows for Smart Cards (WfSC) [1, 16], Smart card.NET [17], and BasicCard [18]. All these initiatives were supported by a number of companies or consortia, claiming that they fulfilled most of the aforementioned requirements. The differences between monolithic and multi-application smart card platforms are illustrated in Fig. 3.1.

There are significant differences, and sometimes misconceptions, on how each of the above technologies is perceived. For example, MULTOS and WfSC are smart card operating systems. Whereas Java Card, GlobalPlatform, BasicCard, and Smart card.NET are considered as platforms that should be located on top of a smart card operating system. Java Card and GlobalPlatform are closely coupled together in terms of how they are deployed, especially in the banking and telecommunications sectors. It is becoming evident that MULTOS, Java Card and GlobalPlatform are the most widely utilised multi-application smart card technologies. For the purpose of this

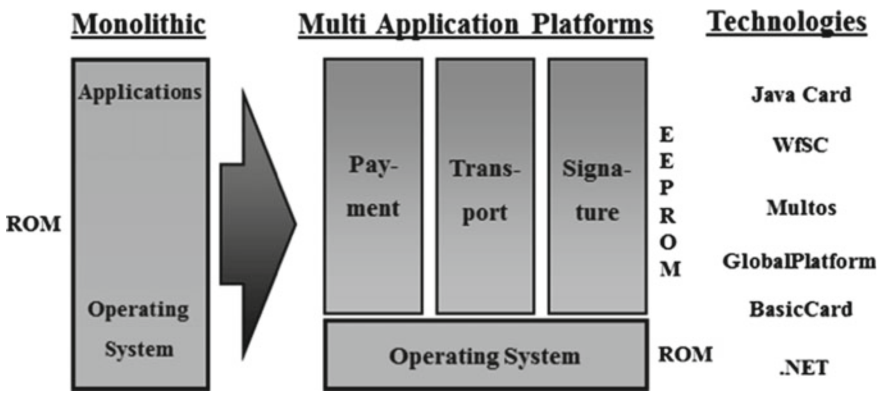


Fig. 3.1 Monolithic and multi-application smart card platforms [6]

analysis, whenever possible the above distinction will be made. However, in general terms we will be referring to both smart card operating systems and platforms.

3.2 Java Card

This section covers issues around one of the most widely utilised multi-application smart card platforms. We begin our analysis by a brief description of the driving force behind Java Card and we then move onto the details of the underlying technology.

3.2.1 Java Card Forum

Soon after the realisation of the Java Card concept (around 1995), a forum was created that would be responsible for promoting and maintaining the Java Card specifications. The Java Card Forum objectives are described in the Java Card Forum web site [19]. In summary, the Java Card Forum aims to promote the Java Card standards and exchange technical information among the participating members by fostering a dialogue that will ensure the "write-once-run-anywhere" concept.

There are different membership levels, and each participating organisation should be committed in promoting the Java Card concept. Current members[1] of the Java Card Forum can be seen at www.javacardforum.com [19].

We believe that among the driving forces behind the creation of the Java Card Forum was the fact that initially each smart card manufacturer was developing its own implementation of a Java Card. Although these implementations had a common denominator, the Java Card language, they were pretty different and more importantly, non interoperable. Therefore, it was evident that interoperability had to be resolved as efficiently as possible.

3.2.2 Java Card Technology

In this section we highlight the main characteristics of Java Card technology.

3.2.2.1 Why Java?

It was around 1990 that Sun began to lay the foundations for a new programming language that would be "hardware independent" and "secure". Although the new programming language was initially called "Oak" [20], it was later renamed to Java [21].

[1]As of August 2016.

Java's development coincided with the growth and wider spread of the worldwide web and due to its fundamental characteristics, it became the preferred programming language for a number applications (mainly in the Internet).

Java is an object oriented language that is closely related to C and C++. However, it does not support pointers and it enforces strict type checking. The actual application can be "sandboxed" (i.e. being executed in a restricted environment) as Java is an interpreted programming language, and therefore any side effects of bad programming are confined within boundaries.

Java programs are written in Java source code. A Java compiler will transform the Java source code into a class file, containing Java bytecodes [22]. The bytecode is processor independent code. These bytecodes are interpreted by a Java Virtual Machine (JVM) [23], which is responsible for executing the program via the native program code of the underlying computing platform. The Java Virtual Machine is the actual interpreting environment that enforces all the necessary security checks and at the same time offers the "write-once", "run-anywhere" concept through the hardware abstraction layer.

One of the greatest advantages of Java can also be characterised as one of its main disadvantages. The fact that Java is interpreted means that it comes at a performance price. This becomes more evident when it is compared with other non interpreted programming languages. However, Java enjoyed wide support from a number of programmers, and as a result a number of development tools and extensive documentation became widely available.

A few of the smart card manufacturers, in their attempt to enhance smart card technology, realised that they needed to overcome the main limitations of existing smart card architectures. This led to the widespread conclusion that the problem faced was similar to that of loading code into the World Wide Web browsers, a problem that Java and other programming languages attempted to solve.

It was soon realised that by using the Java programming language the development of smart card applications is simplified and improved. The main reason was that Java card applications could be written much more quickly since the details of smart card complexities are hidden. The Java programming language paradigm offered a secure model that prevented programs from gaining unauthorised access to sensitive information. This further implied that Java Card applications adhering to the general Java programming model will be naturally confined within their own operational environment. Furthermore, as Java is based on a runtime bytecode interpreter the portability issue is successfully addressed. This implies that Java Card applications would be portable (at least in theory) between different smart card microprocessors. Upgrading the smart card's functionality could take place with new improved applications that could be installed at any time during the life cycle of the smart card. Therefore, the major problem that had to be solved was the migration of the general Java scheme into a smart card environment (taking into account the smart card hardware and memory constraints).

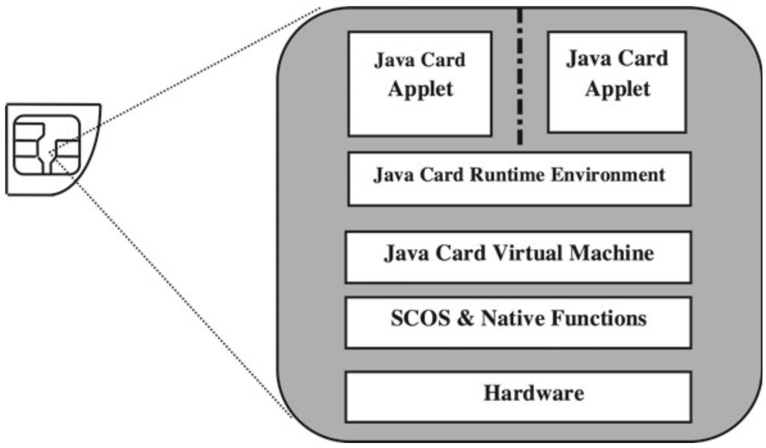


Fig. 3.2 Overview of the Java Card architecture

3.2.2.2 What Is Java Card?

Java Card, as its name implies, should be a smart card capable of executing Java applications. However, the reality is different, mainly due to smart card hardware restrictions. The Java Card language is a subset of the normal Java language. The Java Card Virtual Machine (JCVM) is a subset of the Java virtual machine. Finally, the Java Card Application Programming Interface (API) is a subset of the normal Java API. An overview of the architecture of Java Card is presented in Fig. 3.2.

The Java Card architecture holds limited resemblance to the normal Java specifications. In the next subsection we will investigate how Java Card technology evolved from the very early stages of its conception, up to the most recent proposals. The Java Card specifications can be obtained from the Oracle website [24] free of charge.

3.2.2.3 Java Card Evolution

Integrity Arts, a spin-off from Gemplus, was among the first companies that started (around 1995) to openly discuss issues around multi-application smart card technology and code interpretation. They had developed the Clasp interpreter for the TOSCA programming language [1]. Almost a year later Integrity Arts was bought by Javasoft.

Schlumberger was among the very first companies that announced (late 1996) that it was working on a Java based smart card. This smart card would support a subset of the Java programming language and as a result a subset of the normal Java byte codes. Schlumberger transferred ownership of the Java card specification to Sun Microsystems and at the same time it released (in October 1996) the Java Card Application Programming Interface (API) version 1.0.

3.2.2.4 Java Card API Ver 1.0

The Java Card API Ver 1.0 was the first attempt to bring the benefits of Java to the smart card world. The underlying smart card environment needed to run the Java Card API Ver 1.0 was a 300 KIP (kilo instructions per second) CPU, 12 kbytes of ROM, 4 kbytes of EEPROM and 512 bytes of RAM. The Java Card API supported the following types: boolean, byte and short data, all object-oriented scope and binding rules, all flow control statements, all operators and modifiers, plus unidimensional arrays. The API consisted mainly of the *java.iso7816* package, which defined the basic commands and error codes, for providing a file system and ability to receive and handle Application Protocol Data Unit (APDU) commands [3], as defined in ISO 7816-4.

Following on from that, Schlumberger and other major companies (e.g. Gemplus, Javasoft, etc.) formed the Java Card Forum in an attempt to foster a dialogue between the industry bodies and more importantly to define additional general purpose APIs for those industries (e.g. the financial and telecommunications industries).

3.2.2.5 Java Card API Ver 2.0

Approximately one year later (October 1997), the Java Card API Ver 2.0 was released by Javasoft. This Java Card API offered significantly more advanced features and extended functionality, which was summarised in three different documents:

- The Java Card Virtual Machine Specification that defined the behaviour of the virtual machine, e.g. Java Card supported and unsupported features, how exceptions are caught, etc.
- The Java Card 2.0 Programming Concepts contained information about the Java Card 2.0 classes and how they can be used in smart card applets. Some of the concepts covered were, transaction atomicity, ISO 7816-4 file system, applet life time, etc.
- Finally, the Java Card API 2.0 Specification, which described all the Java Card packages, classes and methods. Among the most notable supported features of the API 2.0 were the following:
  - Packages are used exactly the way they are defined in standard Java.
  - Dynamic Object creation of both class instances and arrays was supported.
  - Virtual Methods and Interfaces, could be defined as in standard Java.
  - Exceptions were “generally” supported.

The requirements for additional ROM (16 kbytes) and EEPROM (8 kbytes) originate from the additional offered functionality. However, due to more adequate optimisation during runtime the requirement for RAM was reduced from 512 bytes to 256 bytes. Furthermore, it was stated that, the Java Card API Ver 2.0 was compliant with the ISO 7816 standard, parts 6, 7 and 8 [25–27].

3.2.2.6 Java Card API Ver 2.1

The next milestone in terms Java Card application development was met in February 1999 when Sun released the latest version of the Java Card API 2.1 [28]. Among the major enhancements and changes from Java Card API 2.0 were the following:

- The applet firewall was more robust and more restrictive. Applets were allowed to communicate with each other through a relatively well defined shareable interface for object sharing.
- The Applet install method was interoperable. This implied that it was represented in a more portable form by a byte array instead of an APDU object.
- A new Exception class hierarchy was defined.
- The Application Identification (AID) was given a more general scope. For example it became possible to compare AIDs.
- The ISO7816-4 file system extension package *javacardx.framework* was deleted. Files were represented as objects.
- The *java.lang* package, was re-defined as a strict subset of Java.
- The cryptography extension package has been reconstructed. This implied that the *javacard.security* and *javacard.crypto* (subject to export restrictions) packages provide extended functionality for security primitives.
- Particular attention was placed in order to improve application interoperability.
- The supported and unsupported features were defined as shown in Table 3.1.

Some smart card manufacturers released Java Card products adhering to the above API, that supported Integer data types and on demand garbage collection.

3.2.2.7 Java Card API Ver 2.2.1

The major improvements within the Java Card API Ver 2.2.1 include the following:

- Support for additional Platforms, e.g. proximity contactless cards.
- Cryptography is expanded, by including Advanced Encryption Standard (AES) and Elliptic Curve Cryptographic (ECC) algorithms.

Table 3.1 Java Card API 2.1 Major supported and unsupported features

Supported	Not supported
† Packages,	† Integer, Float, Double, Long, Char,
† Interfaces,	† String, multi-dimensional Arrays
† Dynamic object creation,	† Multiple "Threads"
† Virtual methods,	† Dynamic class loading
† Exceptions,	† Security manager
† Boolean, byte, short,	† Cloning
† Objects,	† Garbage collection
† Single dimensional arrays,	
† Flow control statements	

- Object Deletion mechanism, e.g. being able to delete applets and packages.
- The API offers the ability to open up to 4 logical channels.
- More functionality, in terms of the Java Card Remote Method Invocation (JCRMI), a long standing Gemplus proposal.

Among the most notable functionality was the Java Card Remote Method Invocation (JCRMI) initiative that was suggested by Gemplus around 1997 [29]. It is a very interesting idea that aims to simplify the way PC/terminal applications interface with smart card resident applications. The terminal application must run in a terminal that supports the Java platform, e.g. Java 2 Platform Standard Edition (J2SE) or Java Micro Edition (formerly known as J2ME). On the smart card the JCRMI server functionality is been executed. From the PC/terminal application, smart card application functionality can be invoked without utilising APDUs, but simply by referencing the necessary object, as if it was an object directly linked with the terminal functionality. See Chap. 10 for more details.

3.2.2.8 Java Card API Ver 2.2.2 Overview

Version 2.2.2 [29] of the Java Card specification introduced several optional and incremental additions to the Java Card Platform. Among the most notable additions are new features for supporting contactless cards. There are also further attempts to include the necessary enhancements to ensure that the Java Card can be as compliant as possible with the Universal Subscriber Identity Module (USIM). See Chap. 4 for more details.

This version of the Java Card standard is fully backward compatible with previous versions. Furthermore, it contains additional cryptographic algorithms (e.g. HMAC-SHA1, SHA-256, etc.), support for logical channels, contactless card support, etc. Key benefits of Java Card 2.2.2 include improved interoperability for cards with multiple communication interfaces, richer cryptography and security features, and standardised biometric support. It also provides a series of new APIs for a more memory efficient application development process.

3.2.2.9 Java Card 3

At the time of writing,[2] Java Card 3.0 is still under development. Some of the most notable suggestions for the enhancement of the Java Card standard include:

- HTTP/HTTPs support, so that the card can be integrated into HTTP/web services environments.
- TCP/IP and HTTP client connections.
- Support for jar files.
- Multi-threading.

[2]As of August 2016.

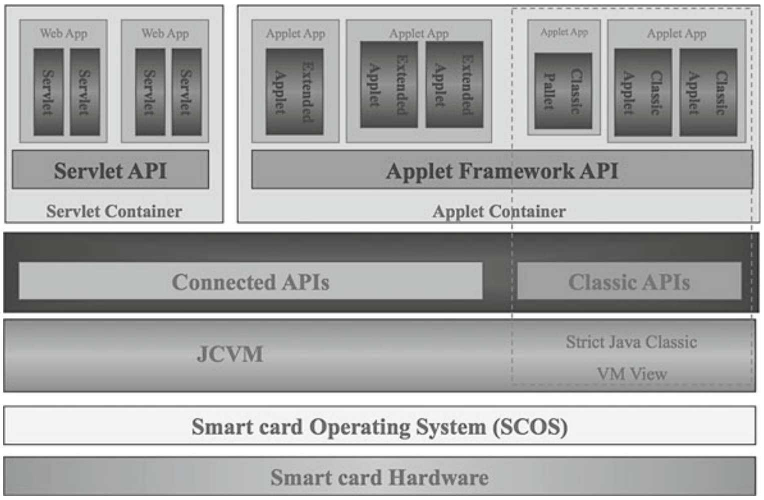


Fig. 3.3 Java Card 3 architecture

- Support for string variables.
- On-card verification of byte code.

The aim of Java Card 3 is to retain the compatibility, interoperability, security and scalability of the previous Java Card 2 versions. It makes the most of advances in hardware and software which have resulted in improved smart card platforms, along with newer protocols and standards, e.g. GlobalPlatform. It allows smart cards to be used as networked components, with new applications such as web services that can run in parallel, utilising multiple interfaces such as Near Field Communication (NFC), contactless and USB. Smart card programming in Java Card 3 is closer to mainstream Java. Typical components and internal architecture of Java Card 3 are shown in Fig. 3.3.

Java Card 3 was released on 31 March 2008 in two separate editions, Classic and Connected.

The Classic edition is designed for use with resource constrained devices, and is based on the evolution of Java Card 2.2.2. Java Card 3 Classic Edition has a number of improvements compared to Java Card API 2.2.2. These include: support for contact and contactless APDUs; off-card and on-card processing of smart card applications; and further cryptographic functionality such as 4096-bit RSA, NSA Suite B (which includes ECC, AES etc.). Java Card 3 Classic continues to rely on the traditionally divided JCVM.

The Connected edition is designed for high-end smart cards with network oriented features that can support web applications in an enhanced execution environment (JCVM). Java Card 3 Connected Edition improves connectivity, and allows the smart card to act as a secure network node, with a Smart Card Web Server (SCWS) that can utilise Java Servlet APIs. Deployment for both traditional and web style

Table 3.2 The four core categories of supported functionality in Java Card 3 connected edition

Function	Notes
Multi-Threading	
On-card bytecode verification	
Automatic garbage collection	More complexity due to networked nature
More utility classes	Char, Long, String
	Multi-dimensional Arrays
	Boolean, Integer and more
	String manipulation classes
	Networking classes connector, connection
	Date and time utility classes calendar, Date, Timezone

applications is supported. Parallel communications are possible, using ISO 7816, contactless, USB, NFC interfaces. Other improved features include on-card byte code verification, automatic garbage collection, multi-threading, and Java class files can now be loaded without pre-processing. Additionally, there have been improvements over transaction atomicity and firewall. There is a more efficient JCVM (version 3)–Connected Limited Device Configuration (CLDC) Specification Ver. 1.1 or the Java Micro Edition, which is intended for 32-bit CPUs. Table 3.2 shows the four core categories of supported functionality.

Java Card 3 Web Applications can interact with off-card clients via HTTP or HTTPs. They are deployed and managed via a web container, i.e. Java Card Platform Web Application Container (JC-WAC). Content can be dynamic or static: servlets are dynamic content, which are computed upon requests from clients and sent back to clients as a response to the request; examples of static content include HTML documents/images/objects residing in files.

Web applications are identified by the specific path, which is rooted within the JC-WAC namespace:

- Path = 1 Application Unique Identifier (A-URI) on the platform
- Path = 2 Application management, inter-application communications and dispatching HTTP requests by off-card clients (i.e. queried URL)

Web containers are able to dispatch multiple HTTP requests concurrently. The Web application deployment descriptor describes the web applications components and how they are mapped into client requests, their security requirements (e.g. authorisation and authentication).

Java Card 3 aims to bring smart card programming closer to Java mainstream programming. This a great step forward but it increases the Java Card complexity and at the same time it may expose smart cards to classical computer security attacks such as: trojan horses; buffer overflows; bug exploits; and existing Java security problems. Therefore, it remains to be seen how soon the current improvements will be finalised and how Java Cards will behave in a widely inter-connected world of Internet-of-Things (IoT) either as secure elements or as standalone smart cards.

3.2.2.10 Java Card Application Development Cycle

The steps for creating and downloading a Java Card application are the following: first of all, the application programmer must take into account the Java Card APIs and write the Java source code for an application. The application can be developed in any standard Java development environment. The compilation process (with a standard Java compiler) will take as input the Java source file and deliver [30] a class file (the application) and an export file. The class file contains the Java bytecode and the export file contains further supplementary information. The Java bytecodes is "nearly as compact as machine code" [4].

Subsequently, the Java Card converter/verifier will have to be invoked and take as input the export and class files and deliver another export file and the CAP file (the interoperable format for the Java Card platform). The converter is an off-card application and it is considered as part of the "off-card" virtual machine entity. The converter performs a number of static verifications in order to make sure that the application conforms to the Java Card API, and adheres to the Java Card framework security requirements. This process is relatively sensitive as its output will be eventually downloaded in the card. Therefore, it is often recommended that the CAP file is cryptographically protected by a digital signature.

In order for an application to be installed, the "off-card" installer takes as input the export file and the CAP file. In cooperation with the "on-card" installer, they perform all the necessary actions, defined by the Java Card framework in order for the application to be downloaded and/or installed. It is often the case that the loading process is independent of the Java Card specification and it is offered as part of other technologies (e.g. GlobalPlatform[3])

In the above procedure, the smart card can be replaced by a software Java Card simulator. In that case the application is downloaded into the simulator and the overall process of smart card application development is simplified and more importantly has reduced development times. This is achieved since the application developer can very easily monitor certain application variables, debug the application and perform the necessary corrections.

3.2.2.11 Java Card Programming Model

The Java Card concept removes the reliance on the underlying smart card operating system and can reside virtually on top any smart card operating system that offers a Hardware Abstraction Layer (HAL). Among the main reasons behind the perceived security that is offered by the Java Card, is the openness and public scrutiny of the Java and Java Card specifications.

In contrast with traditional Java, the lifetime of the Java Card begins when the smart card operating system and Java Card Virtual Machine (JCVM) are burned into the ROM of the card. The lifetime of the Java Card Virtual Machine, unlike

[3]See also in Sect. 3.3.2 GlobalPlatform [14].

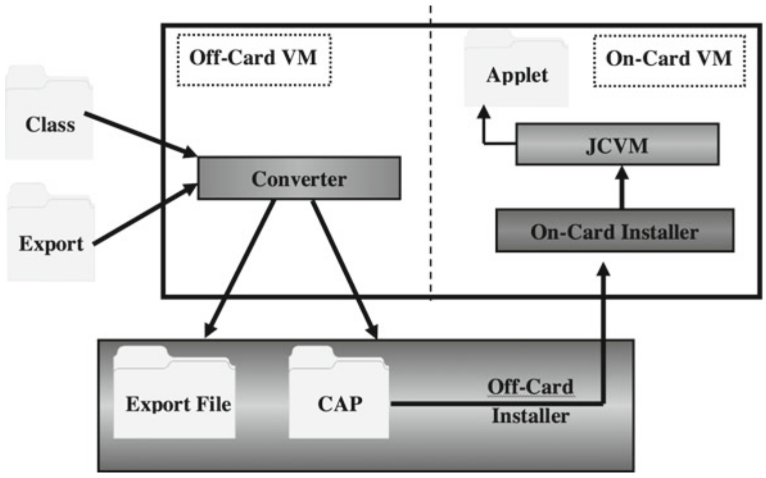


Fig. 3.4 The architecture of the Java Card virtual machine

in traditional Java environments, is supposed to run forever. The architecture of the JCVM is illustrated in Fig. 3.4.

The lifetime of a Java Card application begins when it is properly installed and registered within the card registry. For this reason the Java Card framework offer a number of routines (e.g. *install, select* that control how an application is installed and selected.

Most of the Java Card VM implementations do not offer garbage collection functionality. This means that application programmers must manually reallocate any memory space, which is not used by their application. This forces the Java Card application programmers to think very carefully about how their programs will behave, especially within the limited memory resources of a smart card microprocessor.

The Java Card specification does not define the notion of a security manager as in the traditional Java environment. The VM is indirectly responsible for enforcing a security policy and making sure that applications remain within the predefined boundaries.

Transaction atomicity was among the very early concepts that were defined within the Java Card APIs. Transaction atomicity implies that any updates to persistent objects will be atomic, i.e. "a transaction should be done or undone completely".

Among the main security features of the Java Card model is the firewall. The Java Card firewall is responsible of providing the necessary isolation mechanism between the applications, i.e. avoid any unauthorised communication between applications. Through the object sharing mechanisms and the firewall, applications are allowed to communicate with each other. The mechanism that allows inter-application communication is very strictly defined with the Java Card APIs. Within this concept, objects are owned by the applet that created them.

The notion of the smart card file system, in the traditional ISO 7816-4 form, has been removed from the recent versions of the Java Card API. However, it is perfectly possible to create file objects with user specific classes, e.g. arrays.

Optional Java Card packages are denoted by “x” that is often referred to extensions. The *java.lang* package contains all the basic Java Card classes (e.g. exceptions). The *javacard.framework* includes all the main classes for applet management. The *javacard.security* contains provides access to cryptographic primitives. Finally, the cryptography is often provided by the *javacardx.crypto* class that provides an interface to the support cryptographic operations.

The existence of native methods is not allowed by the Java Card specification. However, this is a partially true statement. A Java Card can have native methods; however, access to these native methods is not allowed/given to the Java Card applets. There is no Java Card API that enables an applet to call these native methods directly. However, Java Card Runtime Environment (JCRE) components can access these native methods–especially when using cryptographic algorithms. “The existence of native methods will completely violate the language based encapsulation model [23] and this one of the main reasons that smart card vendors decided not to offer such functionality to applet developers. The existence of native methods would also compromise portability [31]”.

Interoperability between the different Java Card implementations was a major issue. Every Java Card developer was implementing the Java Card standard in their own preferred way. This resulted in a lot bad publicity for the Java Card. The existence of the Java Card Protection Profile [32] not only aims to provide the necessary security assurance, but it also attempts to provide the necessary cornerstones for making sure that Java Card implementations can be verified under robust evaluation criteria.

It looks like Java is the most widely utilised programming language for smart card application development. This is particularly true taking into account the vast number of issued Java Cards within the telecommunications and banking sector. However, although it appears that Java Card has been promoted effectively, and seems that it will dominate the smart card market, only time will tell whether it will survive in the long run.

3.3 GlobalPlatform

This chapter covers the GlobalPlatform Card Specification as a secure and interoperable multi-application smart card management platform.

3.3.1 The GlobalPlatform Association

During the late 1990s, Visa International, as one of the largest card Issuers in the world, began to explore the issues around multi-application smart card technology.

A couple of years later it created the Visa OpenPlatform (VOP) set of standards. These standards defined how multiple applications could be handled at the card, terminal and smart card management system level. Soon afterwards Visa realised that in order to ensure wider adoption for these standards they had to be as open as possible and publicly available. This along with other business decisions, led Visa to donate the Visa OpenPlatform to the OpenPlatform consortium. The consortium was composed from a number of organisations that possessed an interest in the technology. At that point, the specifications were renamed as the OpenPlatform specifications. Around 1999, as the OpenPlatform consortium was renamed to GlobalPlatform and as a result the specifications were also renamed.

The GlobalPlatform is an independent, non-profit association that is responsible for promoting the GlobalPlatform standards and smart card technology in general. There are different membership levels, leading to different participation preferences and rights. Please see www.globalplatform.org/membershipcurrent.asp for an up-to-date list of GlobalPlatform members. As previously mentioned the GlobalPlatform standards define best practices and architecture for cards and operating systems, terminals, and back-office systems. There are also different committees that oversee the development of the different technologies. The main idea behind the GlobalPlatform standards is to standardise certain aspects of the technology so that interoperability, availability and security of multi-application smart card technology is enhanced. The GlobalPlatform specifications can be obtained from the GlobalPlatform web site www.globalplatform.org free of charge.

It is claimed that "17.7 billion Secure Elements (SEs) based on GlobalPlatform Specifications were deployed globally between 2010 and 2015 [33]." Therefore GlobalPlatform is considered as one of the most widely deployed multi-application smart card platforms. For the purpose of our analysis we will only concentrate on the GlobalPlatform card specification.

3.3.2 The GlobalPlatform Card Specification

In this section, we highlight the main components of the GlobalPlatform card specification.

3.3.2.1 GlobalPlatform Architecture

The GlobalPlatform Card Specification (GPCS) defines a set of logical components that aim to enhance multi-application smart card security, portability and interoperability. The GlobalPlatform Architecture is illustrated in Fig. 3.5.

The GlobalPlatform Card Specification is agnostic of the underlying smart card platform. At the bottom of the suggested GlobalPlatform card architecture we encounter the smart card microprocessor. Usually on top of the smart card hardware we have the Runtime-Environment (RTE). The RTE is considered as an abstraction

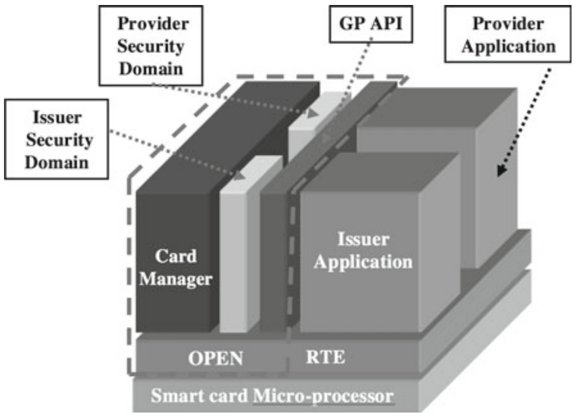


Fig. 3.5 GlobalPlatform card architecture

layer between the GPCS and the underlying hardware. Typically, a RTE is composed of the Smart Card Operating System (SCOS), a Virtual Machine (VM), and an Application Programming Interface (API). For example, in the context of our analysis, the underlying RTE could be a smart card supporting the Java Card specification. In principle, it could be any other smart card supporting any smart card operating system or underlying VM (e.g. WfSC as it will be explained in the next sections).

3.3.2.2 GlobalPlatform Card and Application Lifecycles

The GlobalPlatform Card Specification defines very rigid stages for the smart card lifecycle. Therefore, a GlobalPlatform smart card can be in one of the following stages:

- OP_READY, defines that the Issuer Security domain is installed and fully operational in handling (as the default selected application) all communication (APDUs) to the outside world. Initial keys are also stored in the card in order to allow the Issuer to securely communicate with the card.
- INITIALIZED, defines an irreversible state transition from OP_READY to INITIALIZED. Its functionality is beyond the scope of the GlobalPlatform card specification. It indicates that the card is not yet ready to be issued to the cardholder.
- SECURED, at this state, the security domain keys and security functionality is in place and the card is ready for post-issuance operations. Getting to this stage is an irreversible operation.
- CARD_LOCKED, at this state, the Issuer has blocked access to third party security domains and applications. The card should not be able to function except via the Issuer security domain. The state transition from SECURED to CARD_LOCKED is reversible.

- TERMINATED, the state transition from any other state to TERMINATED (e.g. under a severe security threat or if the card is expired) is irreversible. Most of the card's functionality is disabled and the card should only respond to the GET DATA [14] command.

Transition from one state to the other is only done through the appropriate security domains or applications.

An application can be installed only when the card is not in a LOCKED or TERMINATED state. Following on from the installation a GlobalPlatform application can be in one of the following states:

- *INSTALLED,* indicates that an application is properly installed and linked with all in card components. Keep in mind that such an application cannot be selected yet as it may not be personalised.
- *SELECTABLE,* applications are capable or receiving commands from off-card entities. It is up to the individual applications to define their behaviour when they are at this state.
- *LOCKED,* only an application or security domain with the appropriate privileges can bring an application to this state. Applications cannot be selected and only the card Issuer security domain can unlock the card.

3.3.2.3 GlobalPlatform Application Installation

An application can be introduced into a GlobalPlatform card as an Executable Module (executable code) contained in an Executable Load File (on-card containers of executable modules). These Executable Modules are processed by the GlobalPlatform and underlying RTE in order to obtain a fully functional and installed application. Executable Load Files can be installed either in ROM/EEPROM during the card's manufacturing phase or in the EEPROM at any later stage during the card's lifecycle. The format of the executable load files are stored in the card are beyond the scope of the specification.

3.3.2.4 GlobalPlatform API

The GlobalPlatform API offers application programmers the ability to access the basic functionality defined within the GlobalPlatform card specification. This functionality varies from initiating a secure channel or locking the card as defined above. It also includes some of the functionality (although it has been extended), which was originally defined within ISO 7816-4.

3.3.2.5 GlobalPlatform Security Domains

Security domains are privileged applications. They are the on-card representative of off-card entities. They have the capability to initiate secure channels (by holding their own keys) but also to handle content management functions. GlobalPlatform applications may be associated with and use the services of a security domain. On the other hand, a security domain can also receive information from an application. Therefore, an application provider may communicate with its on-card application via its security domain. There are three main types of security domains in a GlobalPlatform card:

1. Issuer Security Domain. All cards have a mandatory Issuer Security Domain. It is the representative of Issuer.
2. Supplementary Security Domains. They are allocated to individual on-card representatives of application providers
3. Controlling Authority Security Domains are the representative of a controlling authority. Such an entity is responsible for enforcing the security policy on all application code loaded to the card.

3.3.2.6 The GlobalPlatform Card Manager

The GlobalPlatform Card Manager (GPCM) is the central controlling entity in the smart card. It is the ultimate representative of the Issuer. In the early versions of the GlobalPlatform card specification, the Card Manager could be viewed as three distinct entities:

- The GlobalPlatform Environment (OPEN),
- The Issuer Security Domain,
- The Cardholder Verification Method Services.

3.3.2.7 The GlobalPlatform Environment (OPEN)

The OPEN has extensive responsibilities, e.g. to provide an API to applications, perform command dispatching, application selection, (optional) logical channel management, and Card Content management, handling APDUs. Some of this functionality is analysed below. It is stated that these functions shall be implemented by the OPEN, only if they are not provided by the underlying runtime environment.

For example, all communication to the outside world (i.e. through APDUs) are received by the OPEN and redirected to the appropriate on card entities (e.g. application or security domain). Furthermore, the OPEN enforces various card content management operations (e.g. application code verification or application installation certificates). Additionally, OPEN is responsible for performing various security management operations (e.g. locking, blocking and enabling access to the three main entities in the cards, i.e. the card, security domains and applications).

In order for OPEN to successfully accomplish all the aforementioned tasks, it must have access to all the relevant card information. All this information is stored in the Card Registry. Exactly how this information (e.g. card entity life cycles, entity privileges, memory allocation, etc.) is managed or stored is not defined within the GlobalPlatform card specification. The contents of the Registry can be modified only by an authenticated card Issuer invoked action, an internal OPEN action, or by an operation invoked by an authorised application.

3.3.2.8 The Issuer Security Domain

The Issuer Security Domain is mainly used for card content management and for secure communication during application personalisation and application runtime, as described above.

3.3.2.9 The Cardholder Verification Methods (CVM)

The Cardholder Verification methods, refer to CVM management services (e.g. Global PIN), that are only accessible to privileged applications. The most obvious supported CVM is a Global Personal Identification Number (PIN). The CVM has different states (e.g. validated, blocked, etc.) and has strict control management on how it should be used.

3.3.2.10 Card Content Management

This is among the main concepts of GlobalPlatform. It is a very critical operation as in case something goes wrong it may put the card under risk.

The content loading process allows an off-card entity (with on-card representation, e.g. through a security domain) to add content into the card. The following points will have to taken into account in order to better realise what is involved:

- Card content downloading is not allowed when the card is in the blocked state.
- An Executable Load File contains an application's executable code (i.e. Executable Module).
- A Load File (the actual file transferred to the card) contains the Load File Data Block.
- The Load File Data Block contains all the information (applications or libraries) required for the construction of an Executable Load File (i.e. a smart card application) as specified by the underlying platform. Examples of Load File Data Blocks are the Java Card CAP file format and the Windows for Smart Cards OPL file format.

- The Load File Data Block Hash is a redundancy check (i.e. a Hash Function) across the whole Load File Data Block.
- The Load File Data Block Signature (DAP-Data Authentication Pattern) is a digital signature on the Load File Data Block.

Upon the successful completion of the content loading process, an Executable Load File becomes present in the non-volatile memory ready for installation, and an entry is created in the GlobalPlatform registry. Following from that the content downloading procedure can continue into two distinct phases, i.e. the content Loading Phase and the Content Installation Phase.

Content Loading is performed by two commands, which are both processed by the Issuer Security Domain before they are forwarded to the OPEN for further processing:

- INSTALL [for load] command, which serves as a load request towards the card and it defines the requirements of the file to be downloaded.
- LOAD command that is used in order to transport the Load File to the card's non-volatile memory.

Upon the successful completion of the loading process, the Content Installation phase can take place at any later stage during the card's lifecycle. For example, a Load File Data Block can be downloaded in the ROM of the card (during the manufacturing phase) and installed after the card has been issued. Card loading in a GlobalPlatform card can be initiated only by authorised entities (e.g. Issuers or authorised third parties).

3.3.2.11 Issuer Content Loading

Issuers should be in complete control of the card. That means that they should be able to download an application at any stage during the card's lifecycle. When an *INSTALL [for load]* request is received, the OPEN examines whether the same Load File has already been downloaded in the card (this is done by checking the Load File AIDs). If the *INSTALL [for load]* command specifies a security domain (which is to be linked with the application), the OPEN will verify that the security domain exists in the card. Then it will also check whether the security domain has the right privileges and whether it is in a valid state to complete the operations. Finally, a Load File Data Block Hash (on the complete Load File Data Block) is send to the security domain.

Suppose that a smart card application, of approximate size 2 kbytes, is about to be downloaded into a GlobalPlatform card. The whole process will be initiated with the *Install [for load]* command. At this point, we must take into account that typical smart cards have a communication (APDU) buffer for 2^8 bytes. Whereas, if a smart card supports the extended APDUs (ISO/IEC 7816-4:2013) then the size of the buffer may be extended up to 2^{16} bytes. Therefore, a number of LOAD Commands will have to be issued in order to transmit the complete application to the smart card. For each LOAD command, the OPEN will have to verify that there is enough available space in the card.

After the last LOAD command is received, the OPEN will check whether there are any additional checks (Data Authentication Pattern (DAP) verification privileges). The Mandated DAP means that a security domain, usually the Issuer security domain, should always verify a digital signature on the Load File Data Block. This provides the Issuer with the capability to require (through a Mandated DAP in his Issuer Security Domain) that all LOAD requests are pre-authorised (with a digital signature).

3.3.2.12 Delegated Management Content Loading

The concept of Delegated Management allows card Issuers to pre-authorise certain content management operations (e.g. content loading). These pre-authorisations leverage, from card Issuers, the responsibility of managing third-party applications. At the same time application providers are empowered with the responsibility or managing their own applications.

This is actually achieved by using *Load Tokens.* A Load token contains a digital signature (from the card Issuer) on the INSTALL [for load] command. The Load Token is included along with the INSTALL [for load] command.

3.3.2.13 Card Content Installation

Up to now we examined how an application can be downloaded in the GlobalPlatform card. However, an application may also have to be installed. Similarly, to content downloading, all *INSTALL [for install]* commands are processed by the Issuer domain security before they are forwarded to the OPEN for further processing. The OPEN will perform some checks (e.g. available memory, no interdependencies, etc.) and if everything appears to be right, it will install the application and perform the necessary linking with the corresponding security domains, and entries in the GlobalPlatform registry. In case of problems, the OPEN will have to terminate the installation process, return both and an error message and the card to a safe state.

3.3.2.14 GlobalPlatform Secure Channel Protocols

The GlobalPlatform card specification defines a number of secure channel protocols. The notion of a secure channel protocol is used for authentication and subsequent cryptographic protection of the communication between the card and the outside world. Secure channels can be used for all GlobalPlatform sensitive commands (e.g. content loading and installation, CVM management, etc.).

3.3.2.15 GlobalPlatform Summary

GlobalPlatform is a multi-application smart card platform that can work with any underlying SCOS. In fact, there were discussions (in the GlobalPlatform Card

Specification 2.2.2 [14]) on how GlobalPlatform can be used with the MULTOS operating system. The GlobalPlatform card specification is accompanied by the GlobalPlatform Common Criteria Security Target Guidelines [34] (providing information on how a Java Card and GlobalPlatform security target can be developed). Additionally, the GlobalPlatform Card Security Requirements Specification [35] (detailed security requirements) has been made available in an attempt to provide additional evidence on the platforms security.

It appears that GlobalPlatform is taking security very seriously. Great efforts have been utilised to make sure that GlobalPlatform remains an "open" standard (freely downloadable from the GlobalPlatform website) that is offering high standards of multi-application smart card management functionality.

It looks like GlobalPlatform is becoming the de-facto standard in downloading and managing smart card applications. For instance, ETSI GSM 03.19 standards rely on GlobalPlatform for smart card application downloading.

It also looks as if the battle of the giants, between Visa and MasterCard, has been settled, at least to some extent, when MasterCard joined the GlobalPlatform association. Furthermore, within the GlobalPlatform card specification Ver 2.2.2 there are specific references on how some of its functionality can coexist within the MULTOS operating system. It appears that GlobalPlatform card specification is heading along an increasingly successful path.

However, the GlobalPlatform association has realised that looking good on paper is not enough. Therefore, it is putting great effort to make sure that GlobalPlatform implementations are properly tested and that GlobalPlatform developers provide accurate product implementations.

3.4 MAOSCO and MULTOS

In this section, we highlight the main characteristics of the MULTOS smart card operating system. More information about MULTOS can be found in Chap. 17 of this book.

3.4.1 The MULTOS Consortium

The MULTOS consortium is a group of independent organisations that possess a common interest in multi-application smart card standards and in particular for the future development of the MULTOS smart card operating system. Once more, there are different membership fees depending on the level of influence each organisation may require.

MAOSCO is the Secretariat body of the MULTOS Consortium: the current members of MAOSCO can be seen at www.maosco.com [36].

3.4.2 MULTOS Specification

In this section, we present an overview of the MULTOS smart card operating system.

3.4.3 The MULTOS Card Architecture

During the 1990s, Natwest put a lot of effort into the development of the Mondex electronic purse [37, 38]. It is often stated [4, 39] that the MULTOS [13] smart card operating system originated from the Natwest development team in their attempt to provide a secure and reliable platform for the Mondex electronic purse.

The MULTOS name is believed to originate from the terms multi-application operating system. It was designed from scratch by focusing on security along with the specific details of the underlying smart card microprocessors. It is considered among the very few non-military products that managed to obtain an ITSEC [40] E6 Certification, which corresponds to EAL7 under Common Criteria [41] security evaluations.

At an early stage of its development, it was almost impossible to obtain access to the MULTOS specification. This partially explains why MULTOS was not as widely utilised as Java Card. A few years later, an independent consortium MAOSCO was created in order to promote and manage the specifications.

The internal architecture of a MULTOS card is described and illustrated in Fig. 3.6. At the bottom of the architecture we encounter the smart card microprocessor. The MULTOS operating system, offering the basic required functionality (e.g. I/O, file

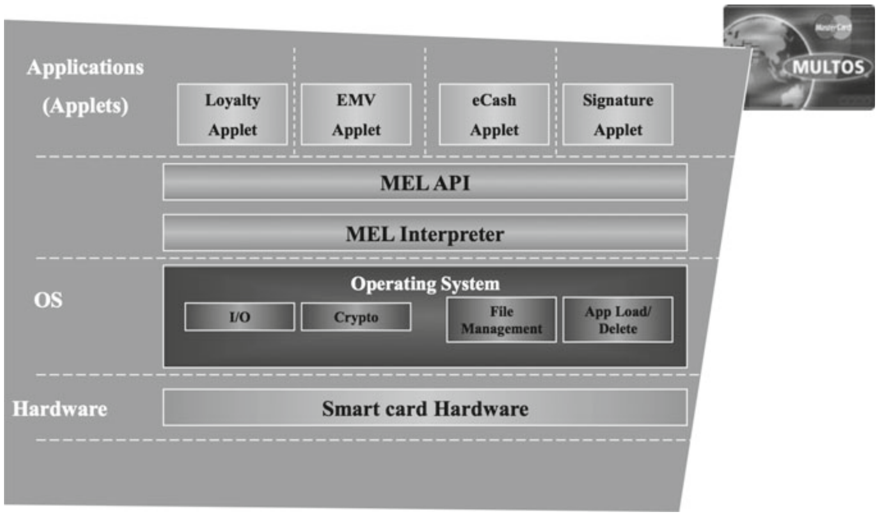


Fig. 3.6 The MULTOS smart card architecture

management, crypto), is located immediately above. This functionality, for example, will allow the MULTOS operating system to dispatch received commands and load and delete applications.

The MULTOS Application Abstraction Machine (i.e. the interpreter) provides the necessary functionality that will enable smart card applications to remain agnostic to the underlying hardware.

3.4.4 MULTOS Executable Language (MEL)

The MULTOS applications were originally written in a byte code assembly language called MULTOS Executable Language (MEL). MEL byte code is the only code that can be executed by the MULTOS operating system. Therefore, all MULTOS applications will have to ultimately be executed as MEL byte codes. According to [42], the MULTOS language is comprised of two languages, numerical one (which is interpreted by the operating system) and the assembly language (which is utilised by the developers).

The MEL language is interpreted by MULTOS and not directly by the underlying smart card hardware, and as a result, a secure environment for application execution is provided. In this way, the applications remain independent from the underlying hardware as any differences are hidden away.

Apart from the MEL byte codes, MULTOS offers a number of operating system primitives that enable programmers to develop low level subroutines that could be called by the applications (even if they are written in high level languages).

MULTOS applications can be developed, using the MEL Application Programming Interface (API), in a number of high level languages (e.g. C, Java) and then compiled into MEL. As already mentioned in the previous types of technologies, an optimiser and off-card loader are utilised in order to download the application into the smart card. The MULTOS application development cycle is illustrated in Fig. 3.7.

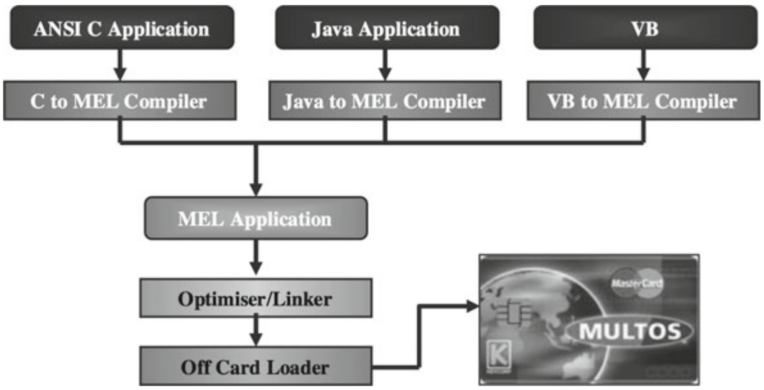


Fig. 3.7 Overview of MULTOS application development cycle

3.4.5 The Application Abstract Machine

As mentioned above, the MULTOS Application Abstract Machine (AAM) is located between applications and the MULTOS OS. It provides an API that will enable application developers to write applications that will be interoperable between different MULTOS implementations. It mainly defines how memory is managed and how applications receive APDU commands and responses.

The MULTOS memory is divided in code space (memory space for the application code) and data space (for all data that the applications may need). Code space and other static variables are stored in the EEPROM memory of the smart card. Each application is provided with its own memory space to hold data and code. Public data and stack data are stored in RAM. As an application will only have access in its own allocated memory, the public data functionality offers the ability for applications to share data. All these memory spaces are handled by different memory registers. It is worth mentioning that the memory space does not relate to physical memory, which is invisible to applications.

3.4.6 Application Loading and Deletion

MULTOS applications (i.e. the actual applications to be downloaded) are contained within Application Load Units (ALUs). The ALUs must also be accompanied by an Application Load Certificate (ALC), which is generated by the MULTOS Certification Authority (CA). MULTOS will only allow applications with a valid ALC to be downloaded in a MULTOS card.

Applications can also be deleted from a MULTOS card with the necessary authorisation. This is obtained in the form of a valid Application Delete Certificate (ADC). Therefore, similarly to GlobalPlatform, the Issuer remains in control of the card as at any stage all downloads and deletion must be pre-authorised.

From the developer point of view, the above strict controls, through the MULTOS CA, imposed a huge burden. However, this situation has changed recently as the MULTOS website (www.multos.com) allows the generation (in a more flexible way) of the above certificates [42].

3.4.7 Communicating with a MULTOS Smart Card

As explained in the previous sections, communication with a smart card is strictly defined within the ISO standards [3]. In a MULTOS card, either the MULTOS operating system will process the command or the command will be redirected to the corresponding (e.g. the default selected) application.

3.4.8 MULTOS Files

The MULTOS files are organised in the file structure defined within ISO 7816-4 [3]. According to this structure, files (e.g. elementary, transparent, fixed, linear and cyclic) are defined in a tree structure with the MasterFile (MF) at the root directory. Therefore, in order to access card data a file must first be selected and then the corresponding command should be executed.

3.4.9 MULTOS Security Features

Each MULTOS application is interpreted by the AAM. Therefore, each application is contained within very well-defined boundaries. At the same time, MULTOS imposes well defined procedures as to how an application maybe downloaded or deleted from the card. Data and applications are controlled through dedicated registers (firewalls) and therefore, undocumented application interference will not be allowed. Actually, this was among the main claims during MULTOS evaluation.

Additionally, the existence of the MULTOS certification authority and the utilisation of a public key cryptography for the load and delete certificates provides a stable and off-the-shelf set of tools and procedures, which is often a major requirement of card Issuers (mainly financial).

MULTOS can be encountered in contact and contactless smart cards. It is mentioned [43] that code size for MEL is significantly smaller compared with other platforms. This can be attributed to the efficiency/inefficiency of the corresponding compilers. The same article also claims that MEL is "in general overall faster" than the other three tested platforms.

A cryptographic co-processor was mandatory in MULTOS implementations. This was deemed necessary in order to perform the necessary application download and delete operations. Although the existence of a crypto coprocessor might improve security and speed, on the other hand it increases the overall cost of the chip.

Implementers of the MULTOS operating system must always adhere to the functionality already defined within the MULTOS standards. This is an attempt to make sure that all MULTOS implementations remain interoperable.

3.5 BasicCard

The BasicCard [18] concept came into existence around 1996 through its inventor Wolfgang Salge and ZeitControl Cardsystems (www.zeitcontrol.de). The following paragraphs provide a summary of the inner workings of the BasicCard platform, which is mainly extracted from [44].

The BasicCard concept relies heavily on the existence and availability of Basic as an easy to learn programming language. The proposed architecture allows smart card programmers to develop applications in the ZeitControl Basic language (ZC-Basic that "resembles many well known Basic dialects such as QBasic but with the necessary constructs to the smart card environment." [44]). The ZC-Basic source code is compiled into P-Code, which is viewed as an intermediate language similar to machine code but only to be executed (at runtime) by a virtual machine. In principle, the same concept is also utilised by the Java programming language although their corresponding intermediate languages are not the same at all. In order to create and download the P-Code to the BasicCard, ZeitControl offers an integrated environment which, at the time of writing,[4] can be downloaded free of charge from the ZeitControl's website. The software allows the debugging and development of BasicCard and associated terminal applications.

At the heart of the BasicCard's operating system we encounter a Basic interpreter (occupying approximately 17 Kb of ROM), which executes P-codes [45]. The syntax of the language contains all the major Basic commands, including strings, IEEE floating-point numbers and various user defined data types. The concept of APDUs exists, but it is completely transparent in BasicCard. For example, the BasicCard programmers simply define their program functionality through functions and procedures and the BasicCard underlying functionality is responsible for translating the necessary calls and handling the APDUs.

The BasicCard allows multiple applications to be downloaded and managed. These applications are associated with access conditions on the corresponding directories (similar or MS-DOS file system) on the card using Access Control Rules (ACRs). In order to strengthen further secure platform management the BasicCard defines a "Secure Transport Mechanism", which allows files and keys to be encrypted with keys shared between the card (loaded during the card initialisation phase) and issuer. Furthermore, the BasicCard defines "an automatic EEPROM Transaction Manager" that allows application developers to specify which file operations and changes to data items occur as an uninterrupted unit. In principle, this is very similar to the transaction atomicity concept in Java Card.

The BasicCard offers a number of additional programming libraries (mainly cryptographic) that aim to support the development of advanced and dedicated applications. Note that there are APIs that allow BasicCard application to be provided in Java and .NET languages. According to Rankl and Effing [4] the BasicCard "program code is very compact and the execution speed is relatively high". ZC-Basic applications can run in a PC without the presence of any additional hardware, e.g. smart card, card Reader. However, among the most widely advertised features of the BasicCard are the low selling prices and by the fact that they can be obtained relatively easily through the ZeitControl website.

[4]As of August 2016.

3.6 Windows for Smart Cards—WfSC

The concept of Windows for Smart Cards (WfSC) was introduced by Microsoft around 1998, but it seems as if the whole concept has been abandoned. At that time, Microsoft realised that as they offered operating systems for PC/server computers and handheld devices they should also be doing something around smart cards. The plan was to offer a version of Windows for smart cards.

The WfSC operating system (requiring an 8-bit CPU, 32 Kb ROM, 32 Kb or EEPROM and 1 Kb of RAM [46]) was presented as a direct competitor to the other multiapplication smart card platforms. But despite great efforts and overwhelming publicity it appears that it never took off and a couple of years later Microsoft withdrew its support.

WfSC came as a fully configurable smart card operating system, as developers had the flexibility to define its exact behaviour. This concept was further enhanced by the existence of a number of functional components offering additional support for a number of industries (e.g. GSM, and additional cryptographic support). The supported file system was based on the ISO 7816 standards [3, 47] and on the well known File Access Table (FAT) of the Microsoft desktop operating systems. Access control to files was enforced by Rule-Based Access Control Lists.

The proposed architecture was subject to security evaluation from Microsoft. Moreover, Microsoft introduced a smart card manufacturer licensing fee for each WfSC that would be developed.

Applications could be developed in VB, C++ and mainly via the closely coupled Visual Studio product. Communication to/from a WfSC card was accomplished by APDUs. The WfSC virtual machine was according to [1] among “the most well designed” of the smart card virtual machines. WfSC, like Java Card, did not specify how applications would be managed, i.e. downloaded, deleted, etc. It was suggested that GlobalPlatform would act as the controlling entity.

3.7 Smartcard.NET Card

Smartcard.NET to bring the functionality of the .NET platform into smart card devices. The actual product name was Nectar Smartcard.NET and it was based on the Microsoft.NET technologies. The platform was developed as close as possible to the international standardisation organisation ECMA335 [48] .NET specifications and the .NET framework.

It is claimed in [49] that following the collapse of the WfSC initiative, Microsoft supported the development of new company that would promote the .NET strategy to the world of smart cards. It is not clear whether this was Hive Minded. It also appears that Axalto worked with Hive Minded to design the .NET card [50]. Eventually Hive Minded was acquired in 2006 by StepNexus, a division of MULTOS International [51] responsible for overseeing the key management authority (KMA) of MULTOS

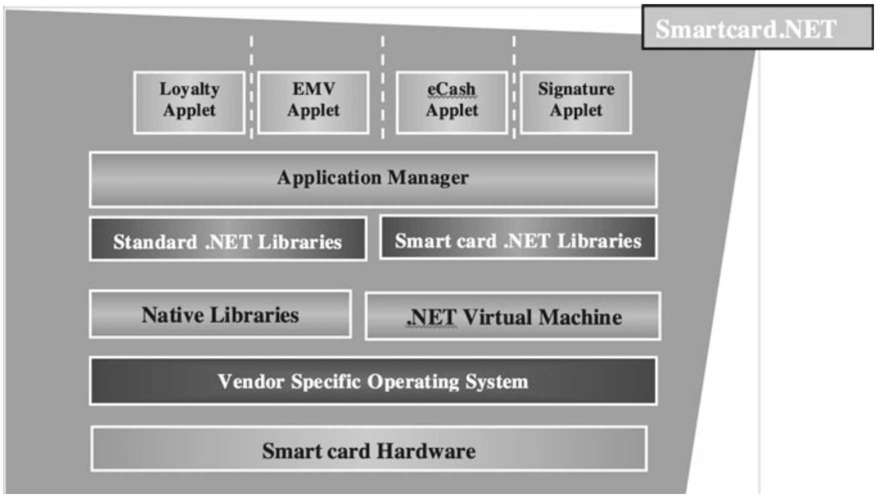


Fig. 3.8 Smartcard.NET architecture

implementations. Currently, it appears that Gemalto (one of the leaders in smart card technology) is offering smart card products based on the .NET architecture [52].

It is defined as a multi-application, multi-language, environment that will offer smart card application interoperability. The targeted application will be compiled into the Microsoft Intermediate Language (MSIL) [53]. Thus, the .NET applications can be written in C#, C++, Visual Basic (VB), J#, JavaScript, etc. that can be compiled into MSIL code. Since the applications written in the above languages will be compiled to the .NET code it becomes possible to combine one or more languages for the same smart card application. The Smartcard.NET architecture is illustrated in Fig. 3.8.

Among the most notable advantages of the .NET platform is that the developer does not need to use APDUs in order to handle the communication with the outside world. Remoting technology allows developers to call on-card application services using TCP/HTTP as the transport layer. The Smartcard.NET card offers a virtual machine that imposes the concept of an "Application Domain" as a mechanism to isolate running applications and avoid undocumented data sharing. The platform also offers a garbage collection mechanism. A remoting mechanism (often referred as RPC in .NET) will enable inter-application communication. Finally, the latest Nectar [54] Smartcard.NET platform also support streams, 64-bit integers and card code verifications. At the time of writing, it was not easy to obtain detailed information about the inner workings of the .NET card. However, a sample figure containing a pictorial view of the architecture is presented in Fig. 3.8. Further information about .NET card technology or related products can be obtained from [55, 56].

3.8 Conclusion

In the above sections, we covered the principal characteristics of the main multi-application smart card platforms. One cannot easily compare one with another as some of them are platforms (residing at the top of an operating system) and some others are operating systems. Furthermore, there are constant improvements to the actual specifications and platforms and therefore indicative comparison factor might not be valid

However, it would be fair to state that the MULTOS smart card operating system was designed by taking security into very serious consideration. It has placed tremendous efforts in order to achieve high levels of confidence through smart card security evaluations. But its closed design, the secrecy around the whole concept and the difficulty in obtaining developments tools and cards, partially contributed towards its restricted deployment. Progressively, the entities promoting MULTOS have realised these drawbacks and attempted to overcome some of the aforementioned problems.

The Java Card programming model is very widely utilised mainly in the banking and telecommunications sectors. The fact that the Java Card specifications were publicly available (along with simulators, documentation and cards) from an early stage, provided a great momentum towards its wider acceptance. Java Card suffered, and to some extend may suffer even today, through interoperability problems and a non verified security model. Both issues have been taken seriously by the smart card manufacturers and a number of Common Criteria Java Cards cards adhering to interoperability standards are in existence.

The GlobalPlatform card specification seems to be becoming the de-facto standard in terms of managing smart card applications in Java Card and perhaps also in MULTOS. From the very beginning it received the necessary attention from the banking industry and recent standardisation efforts [57] will bring it even closer to the GSM industry.

The WfSC operating system has disappeared and it is very difficult to find information around its design or any major companies/industries supporting it. The situation appears to be slightly different with the Smartcard.NET and BasicCard platforms. They are both supported by their corresponding companies and it also relatively easier to find information around their supported infrastructures. However, it remains to be seen whether they will really take off and be utilised in large scale deployed projects that currently use Java Card, GlobalPlatform and MULTOS.

The battle for the winning platform is not yet over!

Acknowledgements The author wishes to thank the sponsors of the Information Security Group Smart Card Centre for their continued support. Also Sheila Cobourne and Danushka Jayasinghe.

References

1. T.M. Jurgensen and S.B. Guthery *Smart Cards: The Developer's Toolkit.* 2002.
2. K. Markantonakis and K. Mayes. An overview of the GlobalPlatform smart card specification. *Information Security Technical Report: Smartcard Security,* 8(1):17–29, 2003. Elsevier Science Ltd (ISSN:1363-4127).
3. ISO/IEC. ISO/IEC 7816-4 *Identification cards - Integrated circuit cards - Part 4: Organisation, security and commands for interchange.* International Organization for Standardization, More Information Available via http://www.iso.org, 2edition, Cited August 2016.
4. W. Rankl and W. Effing. *Smart Card Handbook.* John Wiley & Sons, Ltd, 3rd edition, 2003. ISBN: 0470856688.
5. Andrew S. Tanenbaum. *Modern Operating Systems.* Prentice Hall, Upper Saddle River, N.J, 2001.
6. K. Markantonakis. *Multiapplication Smart Card Platforms*, [PowerPoint slides], 2016. Available via https://rhul.elearning.london.ac.uk/course/view.php?id=210.
7. K. Markantonakis. Is the performance of the cryptographic functions the real bottleneck? In M. Dupuy and P. Paradinas, editors, *Trusted Information: The New Decade Challenge,* IFIP TC11 16th International Conference on Information Security (IFIP/SEC'01) June 11–13, pages 77–92. Kluwer Academic Publishers, 2001. Paris, France.
8. W.A. Ettlin and G. Solber. *Microsoft Basic Book/Macintosh Edition.* McGraw-Hill Osborne Media, 1985.
9. M. Braentsch, P. Buhlier, T. Eirich, F. Horing, and M. Oestreicher. *Java Card - from hype to reality.* Mobile Computing - IEEE Concurrency, October 1999. IBM Zurich Research Laboratory.
10. Sun Microsystems Inc. *Runtime Environment Specification; Java Card Platform, Version 2.1.1.* More Information Available via http://download.oracle.com/otndocs/jcp/7233-javacard-2.1.1-spec-oth-JSpec/, Cited August 2016.
11. Z. Chen. *Java Card Technology for Smart Cards: Architecture and Programmer's Guide.* The Java Series. Addison-Wesley, June 2000. ISBN: 0201703297.
12. MAOSCO Ltd., *MULTOS technology,* More Information Available via http://www.multos.com/technology, Cited August 2016.
13. MAOSCO Ltd., *MULTOS Developer's Reference Manual*, MAO-DOC-TEC-006v1.51. More Information Available via http://www.multos.com/uploads/MDRM.pdf Cited August 2016.
14. GlobalPlatform. *Card Specification v2.2.* More Information Available via http://www.globalplatform.org, Cited August 2016.
15. GlobalPlatform. *GlobalPlatform Card Specification* Version 2.3. More Information Available via http://www.globalplatform.org/specificationscard.asp, December 2015.
16. Microsoft, *Introduction to Windows for Smart Cards,* More Information Available via https://technet.microsoft.com/en-us/library/dd277375.aspx, Cited August 2016.
17. Smartcard Trends. .NET brings web services to smart cards. In *Smart card Trends,* volume 1, page 12. April 2004.
18. Z eitControl. *Basiccard.* More Information Available via http://www.basiccard.com/, Cited August 2016.
19. About the Java Card Forum (JCF), More Information Available via https://javacardforum.com, Cited August 2016.
20. J. Byous. *Java technology: The early years.*, Sun Developer Network, 1998.
21. Oracle Technology Network, *Java 2 Platform Standard Edition 5.0.* More Information Available via http://www.oracle.com/technetwork/java/javase/index-jsp-135232.html, Cited August 2016.
22. B. Venners. *The Java Virtual Machine*, McGraw-Hill, New York, 1998.
23. T. Lindholm, F. Yellin, G. Bracha, A. Buckley. The Java Virtual Machine specification Java SE 8 Edition. More Information Available via https://docs.oracle.com/javase/specs/jvms/se8/jvms8.pdf, Cited August 2016.

24. Oracle. *Oracle*. More Information Available via http://www.oracle.com/technetwork/java/embedded/javacard/overview/index.html, Cited August 2016.
25. ISO/IEC. ISO/IEC 7816-6 *Identification cards - Integrated circuit cards - Part 6: Interindustry data elements for interchange.* International Organization for Standardization, More Information Available via http://www.iso.org, Cited August 2016.
26. ISO/IEC. ISO/IEC 7816-7 *Identification cards - Integrated circuit(s) cards with contacts-Part 7: Interindustry commands for Structured Card Query Language (SCQL).* International Organization for Standardization, More Information Available via http://www.iso.org, Cited August 2016.
27. ISO/IEC. ISO/IEC 7816-8 *Identification cards - Integrated circuit cards - Part 8: Commands for security operations.* International Organization for Standardization, More Information Available via http://www.iso.org, Cited August 2016.
28. Sun Microsystems Inc., *The Java Card API Ver 2.1 Specification*. More Information Available via http://download.oracle.com/otndocs/jcp/7234-javacard-2.1-spec-oth-JSpec/, Cited August 2016.
29. Sun Microsystems Inc., *The Java Card API Ver 2.2.2 Specification*. More Information Available via http://www.oracle.com/technetwork/java/javacard/specs-138637.html, Cited August 2016.
30. D.-W. Kim, and M.-S. Jung, I. Chong, (Ed.) *A Study on the Optimization of Class File for Java Card Platform*, Information Networking: Wired Communications and Management: International Conference, ICOIN 2002 Cheju Island, Korea, January 30 – February 1, 2002 Revised Papers, Part I, Springer Berlin Heidelberg, 2002, 563–570.
31. M. Tunstall, D. Sauveron, K. Markantonakis and K. Mayes. *Smart card Security, volume 50 of Studies in Computational Intelligence,* chapter Studies in Computational Intelligence, pages 205–237. 2007.
32. Oracle Technology Network, *Java Card Protection Profile V3.0*. More Information Available via http://www.oracle.com/technetwork/java/javacard/pp-142498.html, Cited August 2016.
33. GlobalPlatform. *GlobalPlatform Technology Deployed on 17.7 Billion Secure Elements*. More Information Available via https://www.globalplatform.org/mediapressview.asp?id=1241, Cited August 2016.
34. GlobalPlatform. *GlobalPlatform Smart Card Security Target Guidelines.* GlobalPlatform, 1.0 edition, October 2005.
35. GlobalPlatform. *GlobalPlatform Card Security Requirements Specification*. GlobalPlatform, 1.0 edition, May 2003.
36. MAOSCO. *The MULTOS Consortium*. More Information Available via http://www.multos.com/consortium/, Cited August 2016.
37. E.K. Clemons, D.C. Croson, and B.W. Weber. Reengineering money: the Mondex stored value card and beyond. In *System Sciences, 1996., Proceedings of the Twenty-Ninth Hawaii International Conference on*, volume 4, pages 254-261 vol.4, 3-6 Jan. 1996.
38. G.R.L. Higgins. Electronic cash in a global world. In *Security and Detection, 1997. ECOS 97., European Conference* on, page 86, 28-30 April 1997.
39. K.E. Mayes K. Markantonakis and F. Piper. *Managing Information Assurance in Financial Services,* chapter Smart Cards for Security and Assurance. Idea Group Publishing, Information Science Publishing, IRM Press, 2007.
40. ITSEC. More Information Available via https://www.cesg.gov.uk/documents/uk-itsec-evaluation-certification-scheme-uksp01-description-scheme, Cited August 2016.
41. Common Criteria. More Information Available via http://www.commoncriteriaportal.org/, Cited August 2016.
42. MAOSCO Ltd., *MULTOS Developers Guide,* More Information Available via http://www.multos.com/uploads/MDG.pdf, Cited August 2016.
43. J. Elliot. The MAOS trap [smart card platforms]. *Computing & Control Engineering Journal* 12, Issue 1: 4–10, February 2001. ISSN: 0956-3385.
44. BasicCard. *The ZeitControl BasicCard Family*. More Information Available via http://209.68.36.204/downloads/bc_pdf.zip, Cited August 2016.

45. Brian Millier. *Basiccards 101, program your first smartcard.* Circuit Cellar, 164:22–27, March 2004.
46. Peter Johannes. MAOS platforms technical status report. Technical report, Europay International, November 1999.
47. ISO/IEC. ISO/IEC 7816-9 *Identification cards - Integrated circuit cards - Part 9: Commands for card management.* International Organization for Standardization, More Information Available via http://www.iso.org, Cited August 2016.
48. ECMA. *Common Language Infrastructure (CLI) Partitions I to VI.* European Computer Manufacturers Association, More Information Available via http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-335.pdf, 6th edition, Cited August 2016.
49. M. Hendry, *Multi-application Smart Cards: Technology and Applications, Cambridge University Press, 2007, ISBN:0521873843, 9780521873840.*
50. Axalto .NET *Axalto unveils smart card powered by .Net*. More Information Available via http://www.networkworld.com/article/2327491/lan-wan/axalto-unveils-smart-card-powered-by--net.html, Cited August 2016.
51. StepNexus *StepNexus Acquires Hive Minded and .NET-based Nectar and trade; Smart Card Technology*. More Information Available via http://www.smartcardalliance.org/stepnexus-acquires-hive-minded-and-net-based-nectartm-smart-card-technology/, Cited August 2016.
52. Gemalto, *IDPrime .NET*. More Information Available via http://www.gemalto.com/Products/dotnet_card/index.html, Cited August 2016.
53. Microsoft, *Compiling to MSIL*. More Information Available via https://msdn.microsoft.com/en-us/library/c5tkafs1(VS.71).aspx, Cited August 2016.
54. *Hive Minded Delivers Smartcard Platform Based on the Microsoft .NET Framework*, More Information Available via http://www.prnewswire.com/news-releases/hive-minded-delivers-smartcard-platform-based-on-the-microsoft-net-framework-71750142.html, Feb 2004.
55. V. Spáčil, *The Security of .NET for Smart Cards*. More Information Available via http://is.muni.cz/th/208177/fi_m/thesis.pdf.?lang=en, Cited August 2016.
56. B. Fouladi, K. Markantonakis and K. Mayes, *Vulnerability Analysis of a Commercial .NET Smart Card*, Smart Card Research and Advanced Applications, Springer series Lecture Notes in Computer Science, Volume 8419, pp 125–139.
57. Third Generation Partnership Project, *Specification of the SIM Application Toolkit for the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface* (Release 4) TS Reference 51.014. More Information Available via http://www.3gpp.org/DynaReport/51014.htm Cited August 2016.

Chapter 4
Smart Cards and Security for Mobile Communications

Keith Mayes and Tim Evans

Abstract There are around 6 billion smart cards issued every year for use within mobile communications devices. The cards, known as Subscriber Identity Modules (SIM), represent some of the most technically advanced smart cards that have ever been deployed. Their original role was to safeguard the system security by providing authentication and supporting confidentiality, but they have evolved to support additional security and value added functionality. The success of SIMs has been largely due to the efforts of a number of standardisation bodies and indeed the technical evolution and standardisation is still progressing. There have been some exciting developments in the form of Near Field Communication (NFC), high-speed interfaces and high capacity smart cards that could expand the role of the SIM. However, developments within mobile devices and service provision, as well as struggles for dominance in the business environment, will likely constrain the SIM's role and affect its physical implementation, ownership, and management. By convention, SIM is commonly used to represent both GSM and 3G/UMTS SIMs, in this chapter, we use USIM when we specifically wish to refer to a SIM for 3G/UMTS.

Keywords SIM · USIM · UICC · R-UIM · SIM toolkit · CAT · STK · JSR177 · USB · Authentication · Mileage · NFC · Menu · GSM · UMTS · BIP · VAS

4.1 Introduction

One would have to lead a very secluded life to be unaware of at least one successful or publicised smart card application. It is therefore quite surprising that the most successful application of all time, that uses the most technically advanced smart

K. Mayes (✉)
Director of the Information Security Group, Head of the School of Mathematics and Information Security, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK
e-mail: keith.mayes@rhul.ac.uk

T. Evans
Vodafone, London, UK
e-mail: tim.evans1@vodafone.com

K. Mayes and K. Markantonakis (eds.), *Smart Cards, Tokens, Security and Applications*, DOI 10.1007/978-3-319-50500-8_4

cards in widespread use today, is largely unknown to most of its users. The reason is that the users do not handle it as a conventional *card* and indeed may never have seen it, although they may rely on it every day. The smart cards in question are found in every GSM [20] phone and the 3G generation [15] of mobile devices. At the time of writing[1] there are reportedly 4.5 billion mobile subscribers and 6 billion mobile smart cards issued each year. The cards are mainly safeguarding security, but they are also providing customer and network data storage as well as supporting value added services. They are also very well standardised and transferable between mobile devices and the fact that they do all this with the minimum of fuss or trouble to their users, is a good endorsement of standardisation and the use of smart cards in general.

The smart card in a GSM phone is usually called a Subscriber Identity Module (SIM) which gives a strong clue to one of its functions. When an abbreviation has been around for a while it is sometimes given an alternative definition, which is not always complimentary, as in the case of the NTSC television standard, which was referred to as *Never The Same Colour*. Fortunately, the alternative definitions for SIM tend to be complimentary (and accurate), e.g. **S**ecure; **I**n the subscribers hand; **M**anaged [42]. As the SIM was originally designed as a standalone smart card application, the term SIM once referred to both the physical card and the application software on it. However, around the same time as 3G networks were being standardised, it was envisioned that a new form of the SIM may sit in parallel with other smart card applications (e.g. payment, ticketing or loyalty applications). This created two new terms: the smart card physical/low level hardware/software platform when used for telecommunication use is now called a UICC[2] and the application software on it that allows 3G telecommunication access is called a USIM. The *U* comes from the fact that what we now call 3G was originally referred to as the Universal Mobile Telecommunications System (UMTS). The USIM is the evolution of the SIM application and it can replace the SIM even for 2G network access. It is possible to have a UICC with both a SIM application and a USIM application on it; especially if a network operator wants the UICC to work with the widest range of handsets possible. For the most part, the original precise definitions have been forgotten and the term SIM is still used for the smart card you plug into a mobile device. In this chapter we sometimes need to differentiate between SIM and USIM devices and so we will use the term SIM, when we do not.

The SIM has been standardised and in service for quite a long time. The European Technical standards Institute (ETSI) [12] first defined the SIM back in 1990 and the 3GPP standards organisation defined the USIM in 1999. There was a smart card enabled telecoms system that just pre-dated even the SIM; this was the C-NET system that introduced cards in 1988.

The SIM in particular has done a very good job. Before it was introduced some networks were losing many millions of pounds revenue through cloning fraud which

[1]As of April 2016.

[2]This has become a common name, but its original definition was Universal Integrated Circuit Card.

was running at around 15%. Since SIM introduction, cloning fraud has become negligible in networks that use robust algorithms on cards from quality vendors.

Making a reliable and secure solution is not easy, as the SIM must work in a wide range of mobile phones and there are thousands of different models in use. Even apparently, identical makes/models of phones can behave differently across different software releases due to changes, or to additional functionality. Fortunately the SIM design has been comprehensively standardised and detailed conformance tests agreed. Understanding the standards bodies and the relevant standards is absolutely vital to anyone wishing to understand or work with SIMs and so the following is a brief introduction.

4.2 SIM/USIM Standards

There are several standardisation bodies and forums that create and maintain documents used to design and test telecommunications smart cards. Some of the most important bodies are shown in Fig. 4.1.

In the centre of standardisation for telecommunications smart cards, is the European Telecommunications Standardisation Institute (ETSI) and its Smart Card Platforms (SCP) group [10]. ETSI has done some fine work in the area of telecommunications standardisation and the success of GSM can be regarded as largely due to the quality of these standards and the skills of the experts that wrote them. ETSI SCP meets several times a year and is attended by industry representatives, such as mobile network operators, handset manufacturers, UICC manufacturers, smart card chip manufacturers and test equipment vendors.

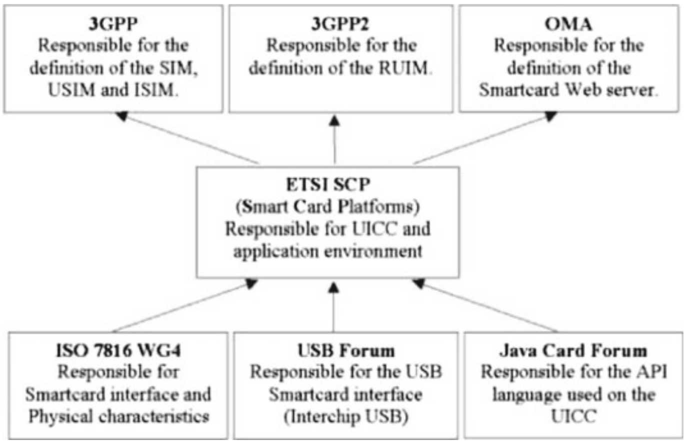

3GPP
Responsible for the definition of the SIM, USIM and ISIM.
3GPP2
Responsible for the definition of the RUIM.
OMA
Responsible for the definition of the Smartcard Web server.
ETSI SCP
(Smart Card Platforms)
Responsible for UICC and application environment
ISO 7816 WG4
Responsible for Smartcard interface and Physical characteristics
USB Forum
Responsible for the USB Smartcard interface (Interchip USB)
Java Card Forum
Responsible for the API language used on the UICC

Fig. 4.1 Smart card standardisation for telecommunications

ETSI SCP has defined the common platform (the UICC) with two types of access (ISO and USB), Secure Remote Management procedures for the UICC, and an Application Programming Interface (API) set for the UICC; they also manage the allocation of identifiers for applications on this platform. To achieve this, the ETSI SCP standards reference other standards created and maintained by ISO 7816 WG4 [16], the USB forum [41] and the Java Card Forum [27]. All ETSI standards are available free from its website.

The Third Generation Partnership Project (3GPP) [28] is a world-wide standardisation body that is now custodian of the GSM specifications and responsible for the further standardisation of the 3G and 4G successor systems, including the SIM and USIM specifications. There is in fact an offshoot of the 3GPP organisation called 3GPP2 dealing with standards for systems that evolved from non-GSM networks; however, 3GPP (or 3GPP1) is the dominant body that we will consider in this chapter.

The International Standards Organisation (ISO) [17] is another important standards body for smart card platforms. It considers the card for general applications such as banking and loyalty as well as telecommunications. For SIMs the most widely referenced standards are the ISO7816 series [16]. The first four standards provide important information on physical/electrical aspects, protocols and inter-industry commands. There are various other ISO standards that become appropriate depending on the particular area of investigation. The biggest drawback to the interested reader is that the standards have to be purchased from ISO and there are strict rules about copying and re-distribution. It is quite easy to purchase the standards via the ISO website; however the costs start to become significant if you need several documents.

The Open Mobile Alliance (OMA) [25] is also worth a mention. OMA evolved partly from the WAP Forum and is now also interested in Multimedia Message Service (MMS) and Device Management (DM). WAP is a Wireless Application Protocol that allows handsets to browse and access mobile services. Within WAP there is the concept of a Wireless Identity Module (WIM) that is a kind of SIM for the WAP world. In fact, the WIM can be incorporated within the SIM and OMA also specifies and makes use of some data files that are stored on the SIM in order to support various OMA services. The specifications can be found on the website.

The Java programming environment is very significant in the smart card world, and smart card applications are often Java *applets*. For a while, it seemed as if Java *midlets* would dominate mobile application development and defacto standards were created to allow midlets to access functionality from the SIM. The Java Community Process (JCP) [18] produced the obscurely named JSR177 specification [19] for this purpose. However, the rapid demise of Symbian phones and the swift rise in Android, Apple and Windows mobile platforms has taken mobile application development into new directions and so the midlet approach is now uncommon.

There are clearly several sources of SIM relevant standards and if you visit their websites for the first time you may be bewildered by the sheer number of specifications and obscure document numbering conventions. To help the reader with some first steps in this area, the following tables identify some “must-have” GSM and 3G

Table 4.1 Common 2G/GSM standards

Standard	Description
3GPP TS 02.17	High level description of the standardised SIM services [29]
3GPP TS 11.11	Detailed technical requirement for the SIM-ME interface. Includes files and procedures (old GSM 11.11) [32]
3GPP TS 11.14	Detailed technical requirements for SIM Toolkit commands and processes (old GSM 11.14) [33]
ISO 7816	Detailed low level (smart card) specification [16]
3GPP TS 03.48	Detailed protocol for the secure transfer of data to/from the SIM and a server in the network (old GSM 03.48) [30]
3GPP TS 03.20	Detailed definition of security related network functions for GSM [31]

Table 4.2 Common 3G/UMTS standards

Standard	Description
3GPP TS 21.111	High level description of the standardised SIM services [34]
3GPP TS 31.101 and 3GPP TS 31.102	Detailed technical requirement for the SIM-ME interface. Includes files and procedures [35, 36]
3GPP TS 31.111	Detailed technical requirements for SIM Toolkit commands and processes [37]
ISO 7816	Detailed low level (smart card) specification [16]
3GPP TS 23.048	Detailed protocol for the secure transfer of data to/from the SIM and a server in the network [38]
3GPP TS 33.102	Detailed definition of security related network functions for 3G [39]

standards. In both cases, we will use the modern numbering format although some of the most well used ETSI references will be noted in passing (Tables 4.1 and 4.2).

Having given a quick introduction to the SIM/USIM, and drawn attention to various standards bodies and important specifications, it is now time to take a look at some of the important things that these devices do for us.

4.3 Subscriber Identity and Authentication

In the introduction chapter, a brief overview was given of SIM authentication and we will now look at this in more detail. The main reason that GSM uses SIM authentication is to securely identify the user account and check that it is valid before allowing access to the network. The mechanism required three items to be available on the SIM card:

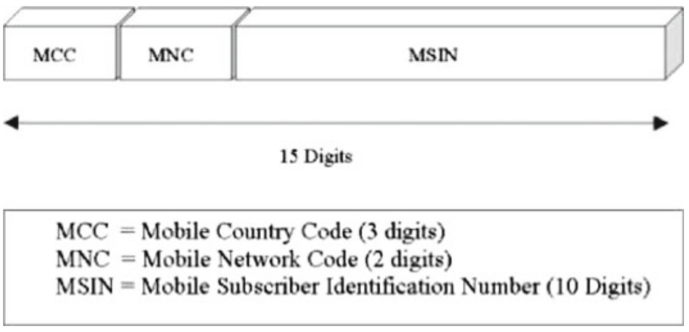

MCC
MNC
MSIN
15 Digits
MCC = Mobile Country Code (3 digits)
MNC = Mobile Network Code (2 digits)
MSIN = Mobile Subscriber Identification Number (10 Digits)

Fig. 4.2 IMSI fields

1. The IMSI
2. The secret Key Ki
3. The authentication algorithm A3

The mobile telephone number of the account is not used as an identifier even though the international format is unique. The IMSI is used instead and only the network operator *should* know how to associate this to the actual telephone number. The format for the IMSI is shown in Fig. 4.2. and the fields uniquely identify a user account anywhere in the world.

Note that to further disguise the user's identity, the IMSI is rarely transmitted in clear and a temporary version (TMSI) is used instead. The IMSI is not a "strong" secret and if the SIM is not PIN[3] locked it can be read direct from the card. Note also that depending on the registration procedures, the network operator might not even know who the actual user/person is, especially with pre-pay SIMs. This is in contrast to say bank cards that are normally personalised to an individual.

The secret Ki is a security sensitive, long-term secret key that has to be stored in an attack and tamper-resistant manner by the smart card. It is the network operator's responsibility (although often delegated to vendors) to ensure that keys are randomly generated for each card. The size of Ki is 128 bits which according to current best practice reports [8], is still acceptable for symmetric key security systems. Getting the algorithm (A3) right is also the network operator's responsibility and it is a common misunderstanding to assume that the algorithm is a standard. What the algorithm has to do, and the messages used to run the algorithm, are standardised, but the actual algorithm is not. The standards are very flexible and allow for all networks to use different algorithms and indeed a single network might chose to support several authentication algorithms at the same time. In reality, attempts have been made to limit the number of algorithms within a particular network, not necessarily because of card complexity but the fact that the algorithms and keys etc. must also be duplicated in the network. The server in the network is called an Authentication Centre (AuC) and this is logically associated with the Home Location Register (HLR)

[3]This has become a common name, but the original meaning was Personal Identification Number.

that is effectively a database holding account-related information. The algorithms tend not to be very processor intensive, as after all they have to run on smart cards, but the AuC must satisfy the requests from the many millions of subscribers using the network without introducing unacceptable delay. How many requests that originate per user can be strongly influenced by operator configuration controls. For example, you could authenticate when you switch on the phone, when you make a call, when you change location area, or perhaps on a timeout basis. Operator policies vary, with some authenticating on a regular and event basis, whereas others may authenticate very rarely. The latter approach weakens security, particularly the confidentiality of ciphered data, which will be mentioned later. A point to note, which is easy to overlook, is that having a good algorithm, long keys plus robust smart cards and servers, may be less effective than you imagine, if you do not have an appropriate authentication policy and configuration.

4.3.1 So How Does SIM Authentication Work?

When the phone powers up, it attempts to read lots of useful information from the SIM. The information is stored in files that may be protected by various PIN codes (more about this later). The IMSI is stored in one of these files and can be read by the mobile. If the user has not enabled his/her PIN lock then there is no restriction on read-access to the IMSI, otherwise the user has to first enter his/her PIN to get the phone to start-up.

Basically, there is a message exchange between the mobile and the server. The mobile sends the IMSI on a radio signalling channel and requests to use the network. The network generates a 128 bit random challenge (RAND) that it uses together with the particular card key (Ki) to calculate a 32 bit expected response (XRES) using the A3 function; it also calculates a 64 bit cipher key (Kc) using the A8 function. The network sends the RAND to the mobile phone that passes it to the SIM card in the form of an APDU containing a RUN_GSM_ALGORITHM command. The command is described in the GSM standards, but summarised in Table 4.3.

Table 4.3 RUNGSM command structure

APDU fields	Contents (hex)	Comments
Class;	A0	Class bytes common to GSM APDUs
Instruction:	88	RUN_GSM_ALGORITHM instruction code
P0:	00	Not-used
P1:	00	Not-used
P2:	10	Number of supplied bytes (in RAND)
Bytes 1–16 (dec)		RAND

Table 4.4 Response to RUNGSM Command

Bytes (dec)	Contents	Comments
1–4	SRES	SIM result from authentication using RAND
5–12	Kc	Cipher key

The response will contain 12 byes of data which includes the 4 byte subscriber response (SRES) and the 8 byte cipher key (Kc), calculated by the SIM in the same way that the corresponding parameters were calculated in the network. The break down of the data field is shown in Table 4.4.

To finish off the GSM authentication part, the mobile sends the results, including SRES, to the network and if it matches XRES the device is authenticated and allowed to use the network. An important principle of the challenge response authentication mechanism is that the secret key (Ki) is never revealed to the mobile or untrusted parts of the network. Furthermore, the functions A3 and A8 should be designed in such a way that it is infeasible to determine Ki even if the attacker can observe or control the parameters RAND, SRES/XRES and Kc.

So far, not much has been said about the cipher key (Kc), but it plays a very important role in ensuring confidentiality of mobile communications. Because of problems with earlier mobile phone systems, GSM was designed to cipher the radio transmissions and so it should not be possible for a third party to eavesdrop the content of calls. As the SIM is a hardware security module one might think that it could cipher the call data, however SIMs were historically too slow for handling real-time call data, both in terms of processing power and data I/O speed. The solution was to build a ciphering algorithm (A5) into all GSM mobile phones which made use of a temporary cipher key Kc (like a session key) from the SIM. This was a reasonable solution as the mobile was much faster and the Kc was changed at each authentication. As discussed earlier, not all operators are as rigorous about authentication as others and that means a Kc may be used for an overly long time. In addition, the key length of 64 bits is considerably shorter than current best practice recommendations (if indeed the network operator is using the maximum key size available). The Kc is also vulnerable in other ways as it might be extracted from the mobile Phone and indeed a copy of this key is stored in clear on the SIM card.

In summary, SIM authentication and its support for ciphering have done a creditable job and there would not be the billions of GSM users and a thriving mobile communications industry if they had not. That is not to say there is no room for improvement and removal of potential vulnerabilities, and 3G standardisation set out to define an improved solution.

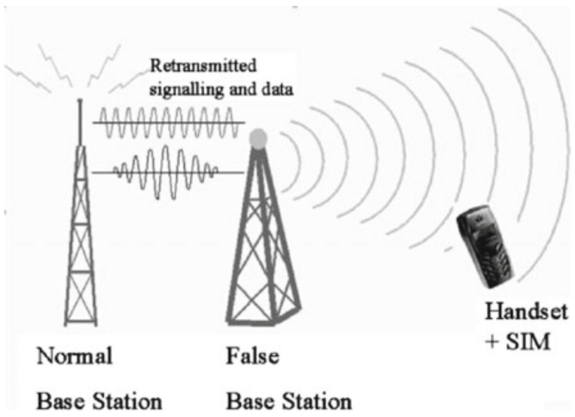


Fig. 4.3 Man-in-the-middle attack on GSM

4.3.2 3G/USIM Authentication/Ciphering

The use of symmetric keys, algorithms and challenge response protocols can also be found in 3G and so the USIM should be considered as an evolution rather than a radical departure from the GSM/SIM approach. One of the potential vulnerabilities of GSM is the fact that only the SIM is authenticated and not the network. The SIM therefore has no way of knowing if the requests it receives are from a genuine source. This can lead to a man-in-the-middle attack as shown in Fig. 4.3.

Basically, the false base station sits between the legitimate subscriber and network, and relays the messages back and forth. It can create, change and suppress messages, so for example, it may convince the mobile that ciphering is not enabled on the network and so voice transmissions could be sent in clear via the fake base station. How widespread this attack has been (if at all) is difficult to say, but with the appropriate equipment it is quite feasible. To remove this potential flaw, the 3G solution uses mutual authentication in combination with a mechanism for integrity protecting critical signalling messages, including the messages used to enable and disable ciphering.

The authentication challenge to the USIM now has a Message Authentication Code (MAC) that has been computed by the network with knowledge of the secret key (K). If the USIM finds that the MAC is correct then the source of the challenge is verified. As a further precaution, a sequence number (SQN) is also provided to prevent an attacker copying and then replaying legitimate challenges. The sequence number has to follow an expected incrementing pattern if the USIM is to consider it valid.

The network calculations are shown in Fig. 4.4. Key K is logically equivalent to the GSM Ki and CK to Kc. You will note that there are two new keys. IK is the integrity key that is used to protect the integrity of signalling messages. AK is an anonymity key used as part of the protocol to disguise the sequence number. AMF is an Authentication Management Field that allows the network to communicate

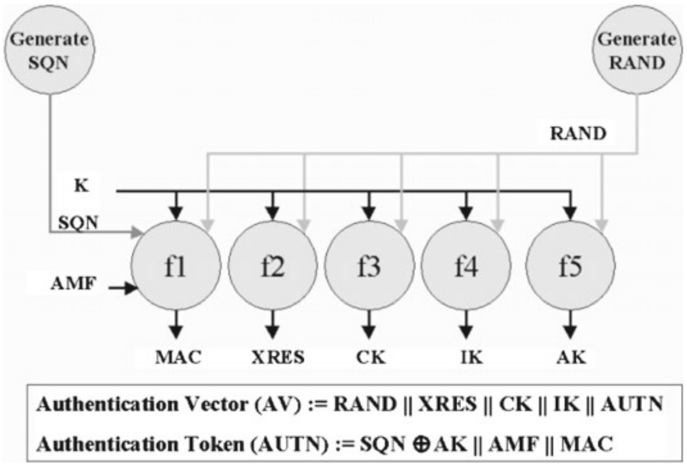

Generate SQN
Generate RAND
RAND
K
SQN
AMF
f1
f2
f3
f4
f5
MAC
XRES
CK
IK
AK
Authentication Vector (AV) := RAND || XRES || CK || IK || AUTN
Authentication Token (AUTN) := SQN ⊕AK || AMF || MAC

Fig. 4.4 UMTS authentication—network challenge calculations

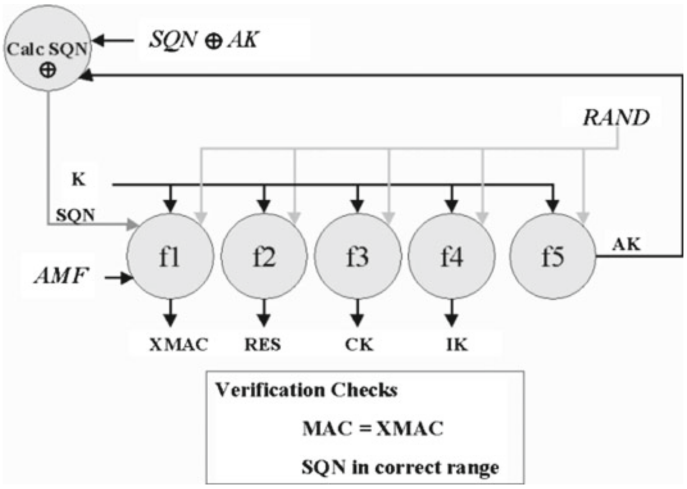


Fig. 4.5 USIM authentication calculations

authentication control information to the USIM. The functions shown in the boxes are defined in the MILENAGE specification [2], which is the example implementation for the 3G algorithm.

The authentication vector (AV = Challenge) consists of five elements (quintuplet) compared to the three element (triplet) used for GSM. The new entries are the Authentication Token (AUTN) to support mutual authentication, management and detect replay attacks plus the Integrity Key (IK). The USIM receives only two of these fields, i.e. the RAND and the AUTN whereas GSM only expects the RAND.

The processing steps taken by the USIM are shown in Fig. 4.5.

Table 4.5 3G authenticate command structure

APDU fields	Contents (hex)	Comments
Class;	00	Class bytes common to 3G APDUs
Instruction:	88	AUTHENTICATE instruction code
P0:	00	Not-used
P1:	81	
P2:	22	Number of supplied bytes
Bytes 01	10	Number of bytes in RAND
Bytes 02–17		RAND
Bytes 18	10	Number of bytes in AUTN
Bytes 19–34		AUTN

Table 4.6 3G authenticate good response

Bytes (dec)	Contents (hex)	Comments
01	DB	Good result
02	08	Bytes in result
03–10		SRES
11	10	Bytes in cipher key
12–27	CK	Cipher key
28	10	Bytes in Integrity key
29–44	IK	Integrity key

The message exchange across the mobile to USIM interface is shown in Table 4.5.

A good response will typically contain 44 byes of data which includes the result calculated by the USIM (SRES) plus the cipher and integrity keys (CK and IK), as shown in Table 4.6.

4.3.3 GSM and MILENAGE Authentication Algorithms

So far very little has been said about the authentication algorithms used in the authentication, other than that the standard defines the mechanism and message exchanges, but does not dictate the algorithms themselves.

There are many GSM authentication algorithms in use and they tend to be kept secret by the network operators. There was an example algorithm for GSM called COMP128 that was made available to members of the GSM MoU Association [14], but was otherwise expected to remain secret. Numerous mobile operators initially adopted the example algorithm, although others preferred their own solutions. Unfortunately COMP128 was weak and relied on secrecy of its design for protection, and predictably, the secrecy could not be maintained. When a design document was

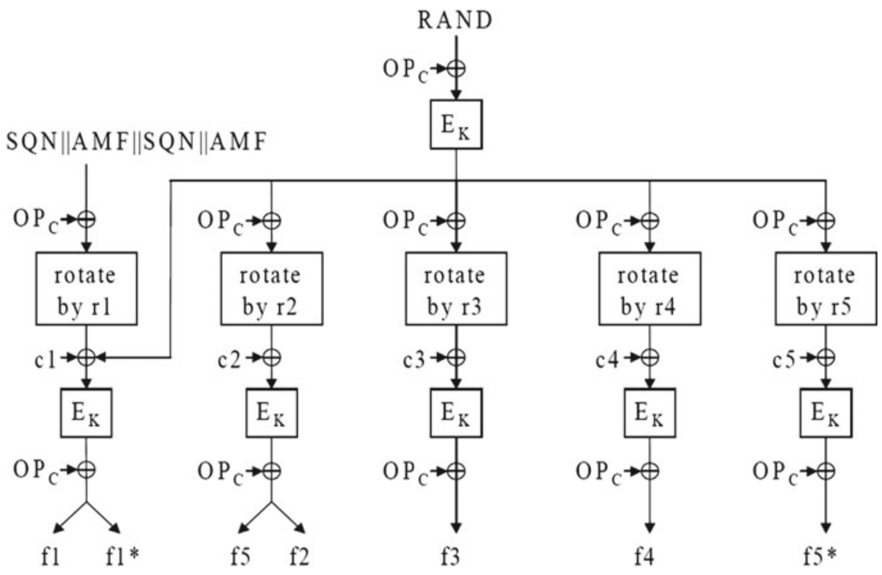


Fig. 4.6 MILENAGE structure

leaked the algorithm was successfully attacked [9] and is no longer recommended for use. Because of the experiences with GSM, the ETSI SAGE [26] group adopted a more open approach with the specification of the 3G “MILENAGE” algorithm. The solution was based around the publicly proven Advanced Encryption Standard (AES) block cipher and it was decided to publish MILENAGE so there was no suggestion that its security relied on the secrecy of the design. The MILENAGE architecture can be seen in Fig. 4.6, which shows how the block cipher (Ek) is used. The OPc is an operator customisation field that is unique for each SIM; pre-computed using a general operator field OP and the individual card key K. The *r* and *c* values are defined in the standard, for fixed rotates and constant XORs, respectively.

For clarity, the essential authentication, ciphering support functions and field sizes used by GSM SIMs and USIMs, are compared in Table 4.7.

4.3.4 The TUAK Authentication Algorithm

The MILENAGE specifications were designed and published by the ETSI Security Algorithms Group of Experts (SAGE). More recently, SAGE specified a second algorithm, called TUAK [3] based on the Keccak [5] sponge function; used in the SHA-3 hash [24]. This was not done because MILENAGE is perceived as vulnerable; indeed it is still regarded as very strong, but rather the need to support Machine-to-Machine (M2M) deployments. M2M systems may for practical reasons require USIM chips

Table 4.7 SIM USIM Authentication Comparison

GSM			UMTS		
Description	Bits	Alg	Description	Bits	Alg
Ki Subscriber auth. key	128		K Subscriber auth. key	128	
RAND random challenge	128		RAND random challenge	128	
XRES expected result	32	A3	XRES expected result	32–128	f2
Kc cipher key	64max	A8	CK cipher key	128	f3
			IK integrity key	128	f4
			AK anonymity key	48	f5
			SQN sequence number	48	
			AMF authentication mgmt field	16	
			MAC message auth. code	64	f1
			AUTN authentication token	128	
Example algorithm					
COMP128-1			MILENAGE		

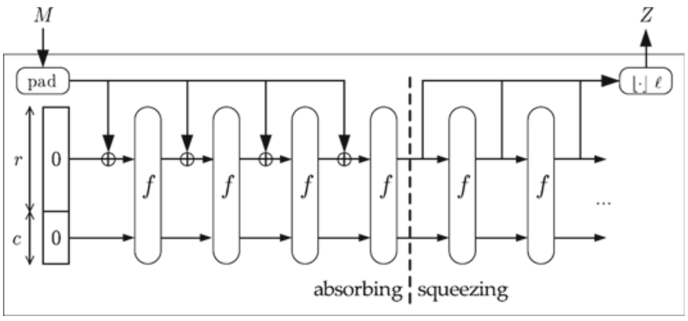


Fig. 4.7 The Keccak sponge

embedded into devices rather than replaceable cards, and the assignment (or re-assignment) to an MNO and the provisioning of security credentials is done later, over the air. MILENAGE as an open/common algorithm will support configuration for different networks, however, given the long operational lifetime of M2M solutions and the potential for vulnerabilities to appear, it is unwise to rely on a single algorithm. TUAK provides the second solution and because it is based on a cryptographic hash function rather than a block cipher, it is unlikely that a discovered vulnerability would be common to both.

The main TUAK building block is the Keccak [5], cryptographic sponge function shown in Fig. 4.7. Essentially data is first written (absorbed) into the sponge buffer, then a fixed length transformation or permutation f is repeatedly applied, before outputting (squeezing out) the result.

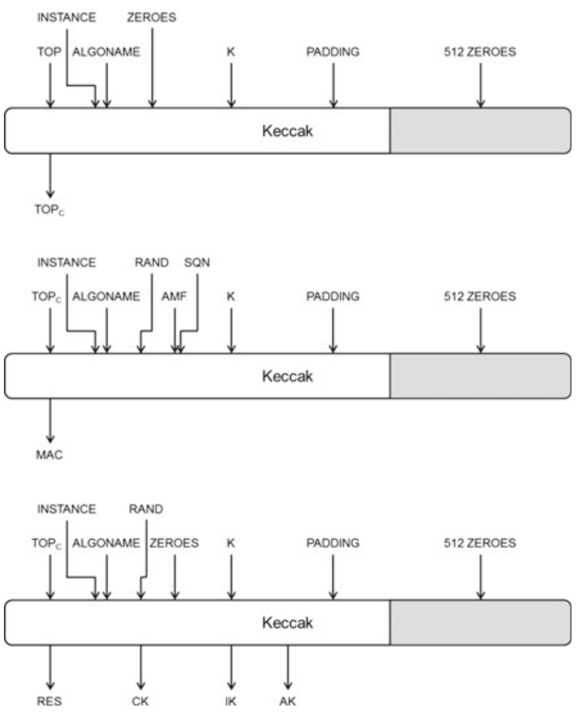


Fig. 4.8 The TUAK algorithm functions

TUAK uses the Keccak algorithm with permutation size n = 1600, capacity c = 512 and rate r = 1088 bits.[4] The capacity relates to the level of security and a 512 bit hash value is comparable to a 256 bit symmetric key size, which satisfies 4G, as well as 3G requirements. The rate relates to how many I/O bits can be processed, and as the TUAK rate is quite large, it is only necessary to run a single instance of the algorithm in all but the most demanding 4G security mode. In Fig. 4.8 the shaded and unshaded buffer areas represent the capacity and rate respectively.

Details of the TUAK algorithm can be found in [3] and performance figures are in [1, 22]. The TUAK algorithm functions are illustrated in Fig. 4.8. TOP_c is equivalent to the MILENAGE OPc and normally pre-computed/stored in the USIM rather than recomputed as shown in the top picture. The MAC is actually the first USIM computation (equivalent to *f1*) as shown in the middle picture. The bottom picture shows how RES, CK, IK and AK are computed simultaneously (equivalent to functions *f2, f3, f4, f5*). INSTANCE is an 8-bit value that takes different values for different functions, for various input and output parameter sizes, and to distinguish between *f1* and *f1**, and between *f5* and *f5**; providing cryptographic separation. ALGONAME is a 56-bit ASCII representation of the string “TUAK1.0”.

[4]Note that these c and r values have nothing to do with the constants used in MILENAGE.

4.4 Just How Secure Is the SIM?

Every so often a story might appear in the press, or perhaps in an academic or hacking forum, which might question the security credentials of the GSM SIM. When such questions are asked we must remember that we have a physical device with attack resistant properties, plus an application (which may include an algorithm) that is designed/chosen to support operational use. If a card Issuer specifies an outdated mode or poor algorithm then that is not the fault of the SIM.

If the security of a credit card is questioned then we can normally refer to a Common Criteria [6] evaluation that will attest to attack resistance of the device, however with SIMs it is rare that we can do this. Although SIMs can be technically advanced and produced by the same companies and factories as produce bank cards, the fast pace and cost pressures of the mobile business rule out formal evaluation. However, this does not mean that SIMs avoid evaluation. Mobile networks typically have a contract clause requiring suppliers to have SIMs independently lab tested against all known attacks; however, no formal internationally recognised Certification is obtained. It is therefore not surprising that there is some suspicion about SIM security and we will look at a couple of examples to filter out some fact from fiction.

4.4.1 COMP128-1 Security

We should know by now that GSM did not standardise an authentication algorithm, but just an algorithm framework. An example (COMP128-1[5]) was accessible to network operators, but some major networks distrusted the example and produced their own algorithms. Comp 128-1 (A3/A8) had inherent design vulnerability and was successfully attacked as long ago as 1998. This was not a SIM, or even a standardisation problem, just a poor choice of algorithm. Perhaps the most surprising thing is that there are places in the world where networks still use COMP128-1, despite the risk of cloning.

4.4.2 Over the Air Update Security

The SIM is a managed platform and because it is within a wireless communications device it is possible to manage it remotely, *Over The Air* (OTA). This is described in more detail within a later chapter, but for now we note that it uses another symmetric key-based protocol to underpin security. In 2013 there was an announcement for a presentation by Karsten Nohl (SRLabs) for the Black Hat Conference in Las Vegas July 2013, in which it was claimed "*This talk ends this myth of unbreakable SIM cards… and illustrates that the cards—like any other computing system—are*

[5]The original name was COMP128, the –1 was added to distinguish it from later improved versions.

plagued by implementation and configuration bugs". Firstly, in the field of information security, it is accepted that nothing is attack/tamper-proof, in fact it can be a trick question in examination papers! A smart card or any other security module should never be described as unbreakable, but rather as strongly tamper-resistant. Secondly, we will see that if the vulnerability exists in practice, it will be the result of multiple poor choices by the network operator. To show why this is the case we will look at the attack in a little more detail.

The aim was to break the OTA security protocol of the SIM to be able to download malicious applications to a Java SIM, then maybe trace the device and the user, send SMSs to premium numbers, divert incoming and outgoing calls, silently include third parties on a call and/or probe for platform weaknesses. The attacker sends a rogue binary SMS message to the phone saying "*here is an OTA application for you*". The binary SMS message includes a signature field, and as the attacker does not know the OTA key at this point, the signature will be incorrect.

A UICC reacts in one of four ways (the attacker needs the last one to extract the key).

- It may reject the incoming message and not send a response—the way recommended by the 3GPP specification since 2012;
- it may send an unsigned response saying "*your signature was invalid*";
- it may send what looks like a signed response saying "*your signature was invalid*", but with all zeroes in the signature field;
- it may send a genuinely signed response saying "*your signature was invalid*";—discarded in the relevant UICC specifications in late 2013.

This is just one of the pre-requisites for the attack to be feasible and we will summarise them all here.

- The SIM must uses single DES for OTA security;—this has been deprecated (advice not to use) in ETSI specifications since 2008.
- The SIM must respond to a fake download request with a specific MAC containing known data—not recommended since 2012.
- The network allows the forwarding of binary messages—likely to be blocked in most networks.
- The OTA implementation does not use a different key for the enciphering of the OTA application or does not encipher it at all—both approaches would be bad practice

Meeting all these pre-requisites sounds very unlikely in mainstream networks, however in a world where some network operators still choose to use COMP128-1 it would be foolhardy to suggest they all make better decisions for their OTA security.

It should be noted that whilst the security functionality discussed above are the fundamental drivers for the use of SIMs, there are many other general functionalities and indeed advanced applications that can be supported by these devices, which will be discussed in the following sections.

4.5 General Added Features

4.5.1 Phone Book

A phone book, or list of useful stored names and numbers has been a much used SIM functionality. In the early days of GSM this was perhaps even more important as you might regularly swap your SIM between say your embedded car phone and your mobile handset. For a long time, the SIM was the default place to store the phonebook, although modern phones usually have an option to use a phone memory. This tends to push the phonebook towards the phone storage, but with corresponding loss of portability and handset vendor independence for the customer. In an attempt to redress this balance, a new phonebook was designed that offers a more complex set of contact details and requires more memory per entry. The basic situation is

User Phone book—Before Release 99.

- SIM storage for three types of entry:-
  - ADN (Abbreviated Dialling Numbers)—choose to call, user can edit.
  - FDN (Fixed Dialling Numbers)—call from list, parent/company to edit.
  - SDN (Service Dialling Numbers)—choose to call, operator edit only.
- Files are stored in a specified location.
- Operator chooses how many of each are available and their size.

User Phone book—Release 99 and after.

- Complex phone book possible which includes over 500 names with email addresses, alternate names, grouping and multiple numbers.
- See 3GPP TS 31.102 [36] for details.

4.5.2 Roaming List

The preferred roaming list is a very important feature that helps determine which mobile network you try to find when travelling away from your home country. Initially this list was designed to speed up the acquisition of a foreign network by directing the phone to networks that the home operator had an agreement with. Generally, when you make a call abroad you are allowed to use a foreign network because your home network has made some kind of roaming deal with them, which involves sharing of the call value. The choice of network may affect the available services, the reliability and quality, but it also has a major effect on the cost of the call to the user and the profit made by the network operators. Deals vary, and the final cost to the user can fluctuate greatly depending on the network selected. The list is meant to help you select the "best" network and is simply a sequential list of networks that you would like to try first if your own is not available. The list contents are initially defined by the network operator and so one would hope that the “best” choices are the most cost efficient for the customer. This is not always the case and the user has the right to modify the list (if he can find the appropriate phone menu), but bear in mind that the network operator is often able to remotely manage and update these lists. As most network operators now have roaming agreements with each other, the list is now used mainly to prioritise networks over each other to get the best level of service for the lowest cost. As these agreements change frequently, it is not uncommon for these lists to be updated remotely by the network operators.

4.5.3 SMS Settings and Storage

When a user sends an SMS message it does not go direct to the recipient, but has to go to a network server/entity, known as a Short Message Switching Centre (SMSC), which sits on the communication signalling network. Each network would normally have their own, and other types of organisation could also have SMSCs. Therefore, the signalling message must be addressed to the appropriate SMSC, however the phone and user do not know the address. Fortunately this information is stored within the SIM card. The SIM can also store the user’s SMS messages, although these days the larger phone memory is more often used.

4.5.4 Last Dialled Numbers

Another useful function is to keep a list of the last numbers dialled by the mobile and a file can be kept on the SIM for this purpose. This is not a mandatory requirement and the phone may be configured to store this list.

4.5.5 Access Control Class

The Access Control Class (ACC) is is an important parameter stored on the card. When a mobile requests access to the network or to make a call, its ACC may be used to determine if this should be allowed. Each network cell lists the Access Overload Control (ACCOLC) classes that it can use and if the ACCOLC entry in the SIM does not match any of the 16 possible classes then the handset will not even attempt to use this cell. This can be used in cases of network congestion when new calls are stopped to manage overloading, but it can also be used to restrict usage for emergency services in the event of an incident.

4.5.6 GPRS Authentication and Encryption Files

In addition to all the authentication functions and data there is also similar functionality to support the GPRS packet data services. The principles are fairly similar and the reader is referred to the standards for a detailed explanation.

The functions listed above make use of information stored in files. The SIM supports a number of file types including Linear Fixed, Binary and Cyclic and these will be described next.

4.6 File Types

The UICC platform uses three file types as defined by ISO 7816 [16]: Linear Fixed, Transparent and Cyclic files. Table 4.8 summarises and compares the standard file types found in SIMs.

Just because a file exists on a SIM does not automatically mean that you can access it. That is because access to files is protected by a hierarchy of PIN codes. A PIN has to be verified by the SIM prior to the particular access taking place. If the verify fails a number of times (typically three) then the PIN may become blocked. Some PINs can be unblocked with a PIN Unblocking Code (PUK), but this is not always the case and some higher level administrative PINs may not even allow one mistaken

Table 4.8 SIM file types

Linear fixed	Transparent	Cyclic
Many records, all are the same length	A single block of data	Many records, all are the same length
READ RECORD command reads	READ BINARY command reads	READ RECORD command reads
UPDATE RECORD command writes	UPDATE BINARY command writes	UPDATE RECORD command writes
Can only be resized by deleting file and creating a new one correctly sized	Can only be resized by deleting file and creating a new one correctly sized	Can only be resized by deleting file and creating a new one correctly sized
Last Record does not wrap to first record		Last Record wraps to first record
Used mainly by the phonebook	Used for most files	Used for last number dialled
Supports absolute record, relative record and SEEK	Can be read/written from any offset (P1, P2)	Supports relative record only

Table 4.9 Verify CHV1 (user PIN) command

APDU fields	Contents (hex)	Comments
Class;	A0	Class bytes common to GSM APDUs
Instruction:	20	Verify PIN command
P0:	00	Not-used
P1:	01	Selects CHV1 (PIN1)
Bytes 01–04		PIN code in ASCII
Bytes 05–08		RFU (some pin codes are longer than 4 bytes)

PIN entry. The user PIN, referred to as CHV1[6] is the most common, although it is often disabled. If disabled or correctly verified, CHV1 allows at least read-access to many of the files on the SIM. The command structure to verify CHV1 is shown in Table 4.9.

A "9000" good response message should result from this and thereafter you and/or your mobile will be able to access files protected by CHV1.

PIN handling can get fairly complex and for USIMs there is the concept of a Global PIN as well as the normal local PINs. The reader is referred to [11] for more information. At this point it is also worth noting that network operators can remotely access the SIM files and so there are also additional mechanisms and keys for this purpose. This is described within the Chap. 11 on Over The Air (OTA) Systems.

4.7 SIMs and USIMs Some Practical Comparisons

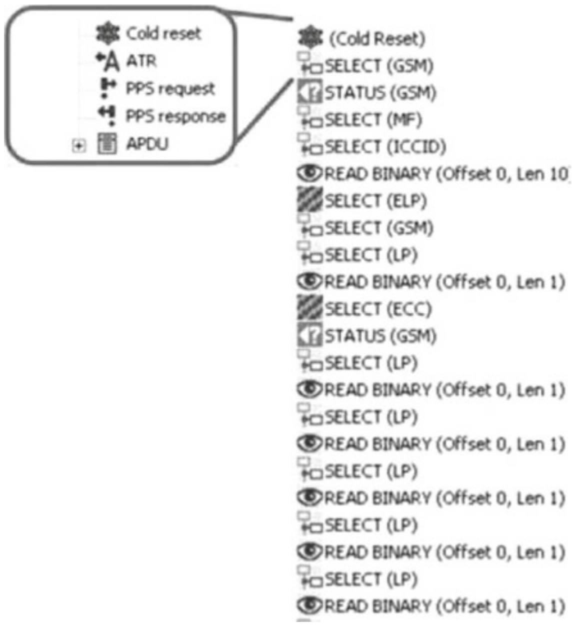
A fairly detailed comparison was given for SIM and USIM authentication, but there are some other practical differences to be aware of. Recall that the SIM was originally defined as a single entity, i.e. there was no distinction between a card platform and the application that ran on it. For USIMs, the situation is different as the USIM is logically an application that runs on a general purpose platform (UICC). In fact the UICC can host other applications including a SIM or some custom functionality. Because there are potentially multiple applications there is a need to select the USIM and this alters the file access hierarchy. This is noted in Table 4.10, where we see that in GSM, one of the first steps is to select the master file as the Root Directory whereas with the USIM we select the USIM application.

It is not perhaps surprising that the SIM standards are now simply maintained rather than improved, with the new features destined for the USIM. Perhaps a more surprising table entry is the one indicating that a SIM can be used for authentication to a 3G network. The standards allow for this to happen and so a network operator might sell 3G handsets with SIM cards. Figures 4.9 and 4.10 show example start-up command sequences for a SIM and a USIM.

[6]Cardholder Verification value 1, or CHV1.

Table 4.10 SIM/USIM usage comparison

Feature	SIM	UICC/USIM
Class byte used	Class = "A0"	Class = "00"
Root directory	MF (3F00)	ADF USIM (7FFF)
Support multiple channels	No	Yes
Authentication command	RUN_GSM_ALGORITHM	AUTHENTICATE
Can be used for GSM access	Yes	Yes
Can be used for 3G access	Yes	Yes
Support SIM toolkit	Yes	Yes
Specified in releases	Ph.1 to Rel.4	Rel.99 to Rel.7
Standards development	Frozen	On-going



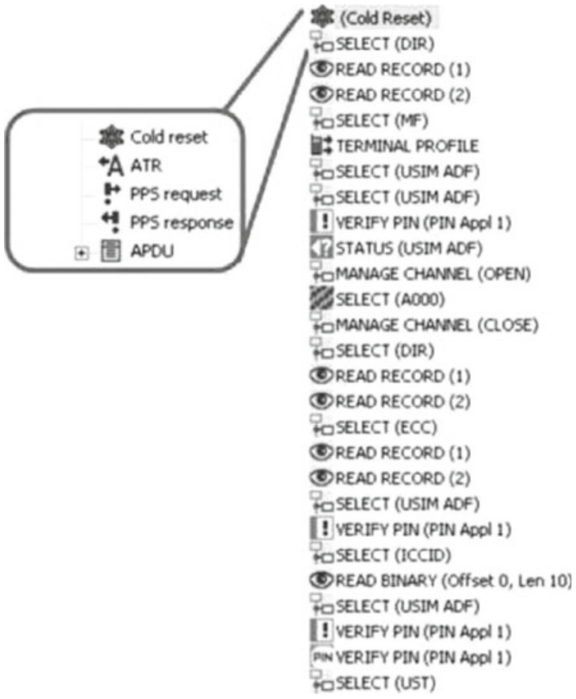
```
Cold reset
ATR
PPS request
PPS response
APDU
```

```
(Cold Reset)
SELECT (GSM)
STATUS (GSM)
SELECT (MF)
SELECT (ICCID)
READ BINARY (Offset 0, Len 10
SELECT (ELP)
SELECT (GSM)
SELECT (LP)
READ BINARY (Offset 0, Len 1)
SELECT (ECC)
STATUS (GSM)
SELECT (LP)
READ BINARY (Offset 0, Len 1)
SELECT (LP)
READ BINARY (Offset 0, Len 1)
SELECT (LP)
READ BINARY (Offset 0, Len 1)
SELECT (LP)
READ BINARY (Offset 0, Len 1)
SELECT (LP)
READ BINARY (Offset 0, Len 1)
```

Fig. 4.9 SIM start-up sequence

Note that in the SIM start-up sequence shown in Fig. 4.9 the SIM is chosen by the use of the 'A0' Class Byte in the APDUs. The terminal will then begin access to the various files by direct selection of the Master File (MF) directory.

By contrast with the SIM, the USIM is communicated to via the Class Byte '00'. The files are now indirectly selected by selecting the USIM application as can be seen in Fig. 4.10. This is a necessary change as we recall that the USIM is designed assuming that it is one of several applications hosted on a UICC platform. The use of additional applications on the UICC needs to be treated with care as traditional platforms are not normally multi-threaded, i.e. only one application runs at a time.


Cold reset
ATR
PPS request
PPS response
APDU

(Cold Reset)
SELECT (DIR)
READ RECORD (1)
READ RECORD (2)
SELECT (MF)
TERMINAL PROFILE
SELECT (USIM ADF)
SELECT (USIM ADF)
VERIFY PIN (PIN Appl 1)
STATUS (USIM ADF)
MANAGE CHANNEL (OPEN)
SELECT (A000)
MANAGE CHANNEL (CLOSE)
SELECT (DIR)
READ RECORD (1)
READ RECORD (2)
SELECT (ECC)
READ RECORD (1)
READ RECORD (2)
SELECT (USIM ADF)
VERIFY PIN (PIN Appl 1)
SELECT (ICCID)
READ BINARY (Offset 0, Len 10)
SELECT (USIM ADF)
VERIFY PIN (PIN Appl 1)
VERIFY PIN (PIN Appl 1)
SELECT (UST)

Fig. 4.10 USIM start-up sequence

The USIM has some real-time duties to perform even when the terminal appears idle and so these must not be adversely affected by running additional applications.

One of the obvious questions is why would anyone want to run additional applications on the SIM? The answer is not hard to find, especially as the extra functionalities are often referred to as Value Added Services. They provide something that has value to the user, which equates to an extra service offering and revenue generation for the network operator. There are of course many ways to provide a mobile service by exploiting network servers and handset capabilities that are often way beyond those of SIMs, but remember our alternative definition for the SIM; "Secure, In the subscribers hand, Manageable". The SIM has proven itself to be a very well standardised and controllable platform that has provided real value-added services across a wide range of legacy handsets.

4.8 SIM Value Added Services

In Chap. 1 a number of smart card application development and implementation routes were briefly mentioned. In this section we will focus on SIM Toolkit. Note that application development is considered in more detail within Chap. 10.

Table 4.11 SIM toolkit commands

User interface	Network interface	Handset interface	Misc.
DISPLAY TEXT	SETUP CALL	PROVIDE LOCAL INFORMATION	TERMINAL PROFILE
GET INPUT	SEND SHORT MESSAGE	POLLING INTERVAL	CALL CONTROL
SELECT ITEM	SEND USSD	POLLING OFF	EVENT TRIGGERING
DISPLAY IDLE MODE TEXT	ENVELOPE (SMS-PP DOWNLOAD)	TIMERS	LAUNCH BROWSER
GET INKEY		MORE TIME	BEARER INDEPENDENT PROTOCOL

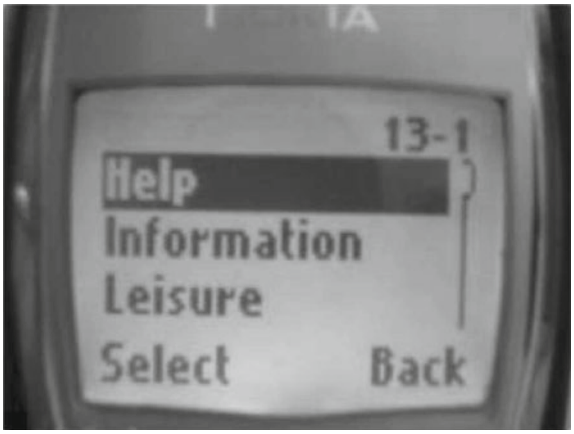


Fig. 4.11 SIM toolkit menu screenshot

We will start with SIM Toolkit[7] (SAT) as it is the oldest and most well-proven route to provide value added services to the user. It is actually quite a departure from the original philosophy of the SIM, in that it was a slave to the handset and merely satisfying its requests. SIM Toolkit turns the tables and allows the SIM to be "proactive" and take temporary control to request the handset to carry out tasks in accordance with applications stored on the SIM. The set of requests and commands that are standardised for the SIM is quite powerful and summarised in Table 4.11.

The most popular SIM Toolkit applications take the form of a simple menu. The user has access to an extra menu that appears to be part of the handset functionality, but is actually fully controlled by the SIM. Text screens are defined via the DISPLAY TEXT commands and the user inputs and menu selections are captured via the GET INPUT and SELECT ITEM commands. A simple example is shown in Fig. 4.11.

[7]May also be referred to as USIM Application Toolkit (USAT) or Card Application Toolkit.

Most menus are geared around making a call, or sending a message to request information, and so they make use of the Network Interface command set. Note that SEND USSD (Unstructured Short Signalling Data) is a fast way to send information that does not need guaranteed delivery and the ENVELOPE command is used for direct server to SIM communications via SMS. The handset can pass some of its operational data to the SIM via the PROVIDE LOCAL INFORMATION command. This is very interesting as the information can include positioning in the form of cell ID, and so enables the SIM to offer location-based services. How often the SIM can get the handset to carry out a command depends on how often it gets the opportunity. The SIM issues its commands via additional responses to every other SIM command. This additional response triggers the handset to fetch the next command. Once the command has been actioned, the handset will send a Terminal Response, indicating the outcome of the command and passing any relevant data. The SIM is only allowed to have one command running at a time so the handset will not fetch another command until the last one has been completed. As the SIM has no internal clock or appreciation of elapsed time, the standards provide for timers hosted by the handset and controlled via the SIM, which can be used to trigger an application after a set interval.

Perhaps one of the most powerful SIM Toolkit commands is CALL CONTROL. The principle is that as the handset is about to set up a call or send an SMS message, it first sends it to the SIM card for checking. The SIM can examine the impending communication action and decide, based on its own functionality and intelligence, whether it is correct and appropriate. The options open to the SIM are enormous. For example, it could simply allow the communications to proceed, it could block the action, or modify it. As an example to illustrate the power of this capability, consider a roaming scenario in which an inexperienced UK traveller takes a foreign holiday and tries to use his/her mobile phone. He/she selects a number from the phone book, but as it lacks the country code, the call fails. The intelligent SIM using CALL CONTROL knows which country it is in (as it is provided by the handset in the CALL CONTROL message) and spots the error and so substitutes the correct international number before placing the call. Event triggering is another way of informing the SIM that something is happening in the handset. It is fast and direct (compared to polling) and allows the SIM to take action. The event could be that the handset has just changed location, or there is an incoming call, or perhaps a call has been dropped.

Bearer Independent Protocol (BIP) is another extremely powerful capability. Normally, most SIM-based applications use the SMS bearer for communicating with network servers, however with BIP you could setup a much faster GPRS or 3G connection. Equally interesting is the ability to set up and control local connections, e.g. via Bluetooth or Infrared, as this then extends the influence and control of the SIM into a whole new area of applications.

Launching a WAP browser may sound a little odd, as historically there has been some competition between the SIM Toolkit/SIM Browsers and WAP. However WAP can provide access to useful services if it is properly configured for the user. Launching the WAP browser from the SIM is a way of ensuring that this will work without requiring the user to manually change local settings and allows an network operator to integrate the two services seamlessly. Furthermore, if the SIM supports the Smart Card Web Server (SCWS) functionality, it could use the launch command to start the browser and point back to a URL hosted within the SIM!

Hopefully the foregoing explanation of SIM Toolkit commands has illustrated just how powerful they can be and why. However, the great disappointment of SIM Toolkit has been the poor and inconsistent support from handsets. This is not the fault of standardisation as the SIM standards are well written and mature, but more due to the decisions of handset vendors on whether to implement the various features. In the early days of GSM one might argue that it was technically difficult to accommodate all the extra functionality, however enough time has passed for this no longer to be the case. One must appreciate that there has always been an uneasy relationship between network operators and handset vendors when it came to the SIM card. The SIM is an embodiment of the network operator's control of security, customers, brand, applications and indeed revenues. The handset vendors on the other hand would like to see their brands more to the fore and more value added applications leveraged and controlled by the handset. The result of this, is that even after many years of stable standards, the most advanced SIM Toolkit features such as CALL CONTROL are not widely supported. However, as the SIM is required to work reliably in any handset that it is placed in, there is a requirement for the SIM to determine the capabilities of the handset and this is achieved via the TERMINAL PROFILE command. Note that the handset will already be well aware of the SIM capabilities from the SIM Service Table that is read during the normal start-up sequence.

Considering the problems with handset support for advanced SIM Toolkit features, one might reasonably question whether SIM Toolkit services have been feasible and successful. The answer is a resounding yes and there have been many business examples that have successfully exploited the core user and network interface commands that are supported on most handsets. SIM Toolkit "Operator Menus" are therefore found on many SIMs. Much of the value comes from making available services more obvious and easier to use and this has a dramatic effect on their usage. If for example a service can be accessed by manually editing code words and sending SMS messages, then changing this to a SIM Toolkit Menu driven service can increase usage by tenfold. If the network operator revenue is taken from SMS messages then the revenue from the service can also increase tenfold. Understandably, companies are rather sensitive about presenting actual service revenue statistics and so Fig. 4.12. is presented as an anonymous example. The graph shows the dramatic improvement in usage of an information service moving from a manual SMS approach to a SIM Toolkit Menu. Of course these days we can construct far more sophisticated services as smart phone applications, but not everyone in the world has a smart phone and there are physical and virtual network operators that can tightly control the functionality of the SIMs they issue, but not the handsets that they are used in.

In smart phone applications there is still a need for an attack resistant security module and there has been interest and some standardisation for using the SIM in this manner, as discussed in the following section.

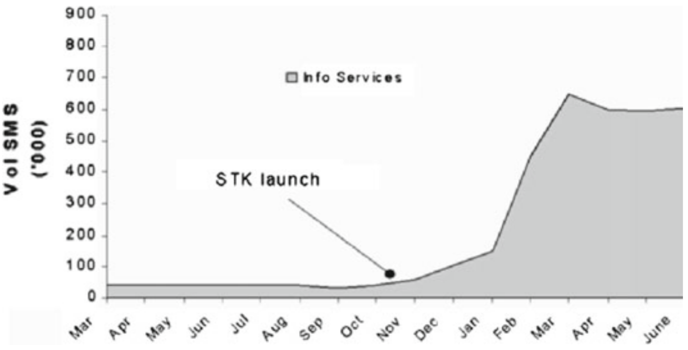


Fig. 4.12 The effect of SIM toolkit (STK) on the usage of a SMS-based information service

4.9 The SIM as a Handset Security Module

An obvious argument against the SIM Toolkit approach is that smart phones are more sophisticated and media rich and so the user service experience (and revenues) can be further enhanced by hosting the functionality on the handset and not the SIM. If the capable handsets represent a large proportion of the deployed base and they are consistent, controllable and reliable, then there is strong justification for this viewpoint.

Sometimes security functionality (such as bulk encryption) has ended up in the handset, simply because the SIM has not had the necessary memory, processing speed and interface speed. These limitations may be challenged by new technology developments (described later), but there seems little business appetite for this, and so increasingly our applications, including the security sensitive ones, will be hosted in the handsets. In this scenario, the SIM can still be of value by providing security functionality and information to the handset.

Handsets are very good at providing user interfaces and have powerful data processing and communication capabilities. However, they have a very poor record when it comes to tamper-resistant security, whereas SIMs are primarily designed as tamper-resistant security modules. A good marriage of capabilities would therefore be to put much of the application in the handset, but exploit the security algorithms and secure storage capabilities of the SIM. It is also important that the interface is "open" to prevent the introduction of proprietary techniques that ultimately would deter developers from using the SIM capabilities. For this reason the JSR-177 [19] defacto standard was written. It includes a set of Java methods (classes) that allow Java MIDlets on the handset to call routines on the SIM. The SIM routines can be called by name or by issuing APDUs and a typical use could be for cryptographic functions such as signatures or encryption. Whilst this sounds positive, JSR-177 has largely gone the way of the best SIM Toolkit features, i.e. there has never been enough support to make it a very practical development option. Smart phone development tools may offer proprietary access to SIM APIs, but little functionality may be available. On this occasion, a lot of the fault lies with network operators, as they have not consistently provided compatible SIMs. Indeed, network operators have missed, or indeed blocked, numerous opportunities to open the security capabilities of the SIM to phone applications, and so it is little wonder that developers look elsewhere for security solutions.

4.10 The Future Evolution of the SIM

Trying to predict the future, especially in the fast moving hi-tech world of mobile communications, is a good way to ensure future embarrassment. This is rather like the old prediction that the UK mobile phone market could be as high as one million subscribers, which at the time was derided as a wildly optimistic target. Nevertheless, there are some SIM developments that must be mentioned, as they could have a radical effect on the future capabilities of SIMs and how they are used within system solutions.

We will start with something that is relatively straightforward (at least at the SIM). This is the concept of adding a R-UIM [4] to the UICC/SIM. To explain what a R-UIM is, requires a brief history lesson. There were two main camps in the 2G mobile communications world, i.e. GSM and IS-95[8] [13]. The former was driven by European standardisation and used SIM cards, whereas the latter was more USA driven and relied on handset security. There were a lot of technical differences between the technologies and so the opportunity for harmonisation only arrived with the 3G standard development. Unfortunately, for political and commercial reasons, it was not possible to have a single world-wide 3G standard, despite a lot of technical similarities. The result is that in addition to the 3GPP standardisation body there is also a 3GPP2 group [40] looking after CDMA2000 (the successor to IS-95). Clearly the multiple standards can present a problem to the roaming user because even if the handset could cope with the various radio interfaces, the user authentication would only work on the home standard. Fortunately CDMA2000 has the possibility of authentication via a smart card-based application called the R-UIM. As the UICC found in a normal 3G phone is a multi-application platform, it is possible to have USIM, SIM and R-UIM applications present. Therefore at the UICC level a roaming solution appears possible, however it should be noted that the more difficult problems have to be solved at the network signalling level.

The idea of adding additional authentication handlers does not need to stop with the R-UIM. In principal you can authenticate to almost anything and the linkage to the SIM means that you may also have a simple billing solution via the normal mobile phone account. If the applications have a strong linkage to a user's identity (which is not necessarily the case with SIMs) then a whole range of identity management options also present themselves.

[8]Sometimes referred to as CDMAOne.

One of the most interesting and exciting developments is the support of Near Field Communications (NFC) [23] in mobile phones. In smart card jargon this is a contactless smart card/RFID interface for handsets. One of the reasons to get excited about this is because it enables the handset to behave as if it is a contactless smart card (or card reader). At the time of writing we have a rapidly expanding set of contactless card services, including passports, health cards, travel tickets and touch and pay purchasing. The NFC phone should use a Security Element (SE) when emulating a smart card and the NFC standards provide an option for this to be within the SIM. However, this is not the only option and the Apple Pay service uses a hardware SE chip built into the phone, whereas the Android Pay equivalent seems to use Host Card Emulation, which is basically a software functional equivalent.

There are lots of ideas for smart card-based services and the business cases may easily support the deployment of smart cards to the customer base. Where the propositions grind to a halt is in the deployment of the reader infrastructure. Even when free smart card readers are sent to end-users they are very rarely capable or indeed inclined to install and use them correctly. However, if you have an NFC capable mobile Phone you are carrying around an advanced reader system with sophisticated user interface, remote communications interface and embedded security module (SIM). Not only does it mean that some smart card service ideas may be resurrected, but the NFC reader handset can interact with all kinds of contactless devices including e-tickets, bank cards, passports and RFIDs. The opportunities from reading RFIDs are not to be underestimated as they will find themselves in all sorts of everyday objects and products.

For example a stranger arrives in town and sees a poster that says “CINEMA”.

- He/she holds his phone against the poster and reads the embedded RFID.
- The location is known either directly from the RFID identity or the phone positioning and triggers the phone to browse the website of the nearest cinema.
- The user selects a film.
- The user pays, either from his/her phone account or holds his/her contactless Payment Card against the handset.
- The service provider downloads an e-ticket to the phone (Virtual RFID)
- User is guided to the cinema using mapping software
- At the cinema the user holds his/her phone against an access gate and as the e-ticket/RFID is valid, is allowed in to watch the film

The possibilities for NFC appear endless, but it is important to note that there is no guarantee that the SIM will be at the heart of it, other than from securing the cellular communications bearer. There are other contenders for the NFC Security Element and this is not used at all in reader mode. Furthermore, the SIM has been kept a closed platform by the network operators, so developers will look to phone-based security functionality, even if less attack resistant compared to the SIM.

Technology is not the critical constraint on the SIM, in fact it could do far more than it does if operators wanted to pay for it. For example, it is quite practical to offer fast High Density/Capacity SIMs [21] to revolutionise the way that these devices are used.

When deciding how to partition a system solution across servers, handsets and SIMs, careful consideration is given to the capabilities and limitations of the various elements. Normally you would not tend to use the SIM for bulk data storage or fast real-time processing. Typically, the SIMs have been small compared to memory cards and at the time of writing[9] a reasonable SIM card might have about 128 kilobytes EEPROM or flash data storage, whereas a non-secure off-the-shelf flash memory card can be about 128GB. One of the reasons that the SIM has remained small is because the chip size is restricted due to some ISO bend and twist tests, but it is also strongly influenced by cost. One could argue that these are not fundamental reasons for avoiding larger SIM card chips. Firstly the modern SIM is never used as a card as it spends its life in a slot within the handset and so providing the card can be delivered and installed without damage, there is no need for the ISO restriction. Secondly if you need a lot of memory, you have to pay for it somehow, whether it is in the SIM, embedded in the phone or a separate flash card. There may be differing production costs but there is also an element of "who pays?" rather than "how much?".

Some things that are stored in mobile devices have little importance, but increasingly there is valuable multimedia such as music files and pictures as well as business, personal and private information. The prospect of storing this information on a security device has some appeal both from a user and content provider perspective. In response to this, some smart card and memory card companies have proposed and indeed demonstrated Gigabyte SIM cards. However, increasing the memory alone does not help matters greatly as the conventional ISO 7816 interface between the handset and SIM is far too slow to allow the extra memory to be exploited. Therefore, a high speed (USB) [41] interface is also implemented. The combination of large memory and fast I/O really challenges the pre-conceived ideas about what a SIM could or could not be used for. It can also have an impact on data encryption. Conventionally the SIM and its interface is too slow to cipher/decipher real-time data such as digitised audio and so this is performed by the handset using a temporary session key produced by the SIM. However, if the interface and the SIM processor is fast enough, the ciphering and deciphering could be performed within the tamper-resistant device. This would probably require more power to be drawn from the handset , but then the handset has to work less hard.

Whether we will actually see the widespread deployment of these high capacity and high performance SIMs is highly unlikely. It depend on handset support, which is largely controlled by the manufacturers who have their own agendas. The network operators still have strong influence, but after years of driving down the costs of SIMs it is unlikely that they would consider paying extra for the high capacity version.

Adding a Smart Card Web Server (SCWS) to a SIM is a nice idea, especially in the context of the Internet of Things (IoT), as all the SIMs become part of an IP network. This is perfectly feasible and has been demonstrated by smart card vendors. Putting an IP stack on the SIM is a natural thing to do as the old ISO 7816 interface was never designed for high performance communications. Indeed, the SIM operating systems and file structures would probably also become outmoded and need redesign.

[9]As of April 2016.

Table 4.12 Newer UICC features of ETSI SCP Rel.7

Feature	Description
High Speed Interface	This adds an 8 Mb/s USB channel, based on USB, which will eventually replace the ISO interface. Will have an IP layer
Contactless Interface	Allows the UICC to act as a contactless smart card via the terminal
Secure UICC—Terminal Interface	Secures the terminal to UICC interface so secure transactions can take place such as DRM
Smart Card Web Server	Specifies how the SIM card can act as a web server

At this point, it is probably worth adding a word of caution. The reason that SIMs have been very successful at protecting mobile communications is that they are restricted and highly controlled platforms. Once the SIMs start looking like IP platforms or indeed PCs then they may inherit their security problems. The key to avoiding these problems is to have well defined standards right from the outset. Fortunately, the issues are already being dealt with by ETSI and at the time of writing the features shown in Table 4.12 are being standardised.

Note that the Secure UICC Terminal Interface is not so much a new service, but a security improvement that will make other services more feasible. Although physical access to the UICC to terminal interface is a little tricky, it is not impossible and so one must assume that the protocol transmissions can be eavesdropped. Indeed it is possible to buy powerful test equipment that performs exactly that purpose [7]. For normal SIM interaction this does not matter too much, but for Digital Right Management (DRM) applications it might be a route to discover a content decryption key that could be used to break copyright protection measures. Encrypting the transmissions is a solution to this but may shift the point of attack from the interface to the handset itself.

4.11 The Future Evolution of Smart Phone Security

Mobile devices have evolved much faster than the SIMs that they host; so predicting the future of smart phone security is more uncertain than SIM capabilities. In fact, since the first edition of the book, the mobile device has become very much more complex from a service and security viewpoint as illustrated in Fig. 4.13.

The mobile device now has multiple wireless communication bearers. Cellular is no longer just GSM as we also have 3G/UMTS and 4G/LTE. Most phones also use WLAN and Bluetooth and increasingly they support Near Field Communications (NFC). There is usually a physical interface for connecting to other devices and sometimes a memory card slot/interface. The SIM is still at the heart of cellular

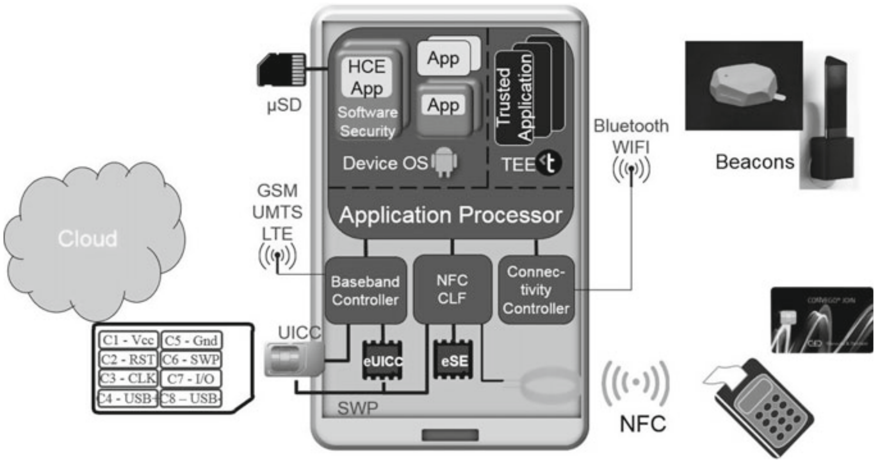


Fig. 4.13 Smart phone security

bearer security, and can be used with the other wireless bearers too, although the latter has other contending solutions. Smart phones that support NFC may have an embedded hardware security module in the form of a Security Element chip, and a modem device might have an embedded chip that could in principle be configured and personalised to act as the SIM.

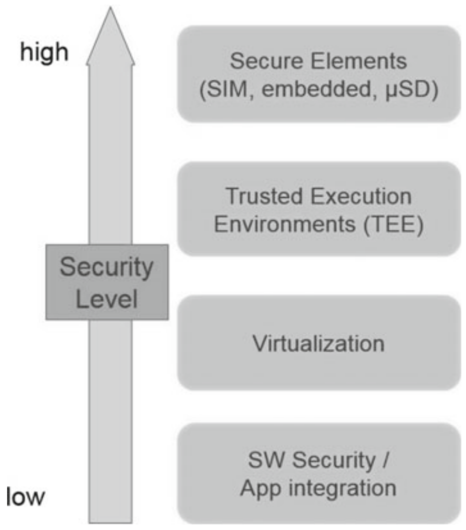
Generally, the security of the main phone processors has been viewed with distrust, and so the use of Host Card Emulation (HCE) in which the NFC SE is simulated in software is not a preferred security solution. However, application security is a growing general issue in mobile devices and there are a number of initiatives to create an isolated Trusted Execution Environment (TEE) for running security sensitive tasks. TEEs supported by processor hardware extensions are likely to evolve and be used increasingly for security applications, however it is unlikely that they will achieve the range and levels of attack resistance assurance that are possible with smart card chips. The fragmented and often proprietary approach to TEE implementation does not help with its development and adoption.

Another approach, more common in servers and PCs, is the use of virtualisation, so that an application's operating system (OS) may be one of several sharing the same physical processor. An application would rely on the virtualisation layer to separate it (and its OS) from all others; as an alternative to relying on the capabilities of one OS to isolate all the applications loaded on it.

When security hardware is not available or accessible to developers they have little option but to fall back on software only defensive techniques that are inferior to hardware defences and can slow execution by orders of magnitude. Figure 4.14 illustrates how we might currently rank the security of different execution environments.

We put specialist hardware security chips at the top, because they are well-proven and simple devices that can be security evaluated; resisting logical, physical, side-

Fig. 4.14 Execution environment security levels


high
low
Security Level
Secure Elements (SIM, embedded, µSD)
Trusted Execution Environments (TEE)
Virtualization
SW Security / App integration

channel and fault attacks. TEEs are next on the list because they have some hardware support for security isolation; although they generally do not attempt to offer protection against physical, fault and side-channel attack; and are probably too dynamic and complex to be confidently evaluated. Virtualisation is above simple application software measures, due its isolation capabilities, although critics will point out the following flaw in the logic. A modern OS is not trusted to maintain security/isolation because it is too large and complex; in other words, the Trusted Computing Base (TCB), the software you must rely on, is just too big. A virtualisation layer could, therefore, help if it was very small, but unfortunately they can be nearly as large as the operating systems they host.

Interestingly, if the figure had instead represented scalability or accessibility to security capabilities, the arrow would be the other way up. The higher the security protection, the more difficult it is for developers to get access and for users to have compatible devices.

Security is not just about devices, and management permissions and processes play a big part. We mentioned earlier that M2M devices may contain eUICC chips that need to be personalised and even re-personalised outside of a secure physical environment and perhaps after issue to the customer. Although OTA techniques can be used to modify security chips, the keys and credentials are normally under the control of one party (the Issuer), but that role is questionable when the network is not known from the outset. A more general ecosystem is needed as shown in Fig. 4.15.

The approach relies on all parties trusting a Subscription Manager (SM). A SM is basically a broker between MNOs and eUICC vendors, and trusted for securely provisioning MNO subscriptions on the eUICC. A complication is that to personalise the chips requires initial wireless communication, so default subscriptions are required for which the SM has full management credentials; as a conventional Issuer

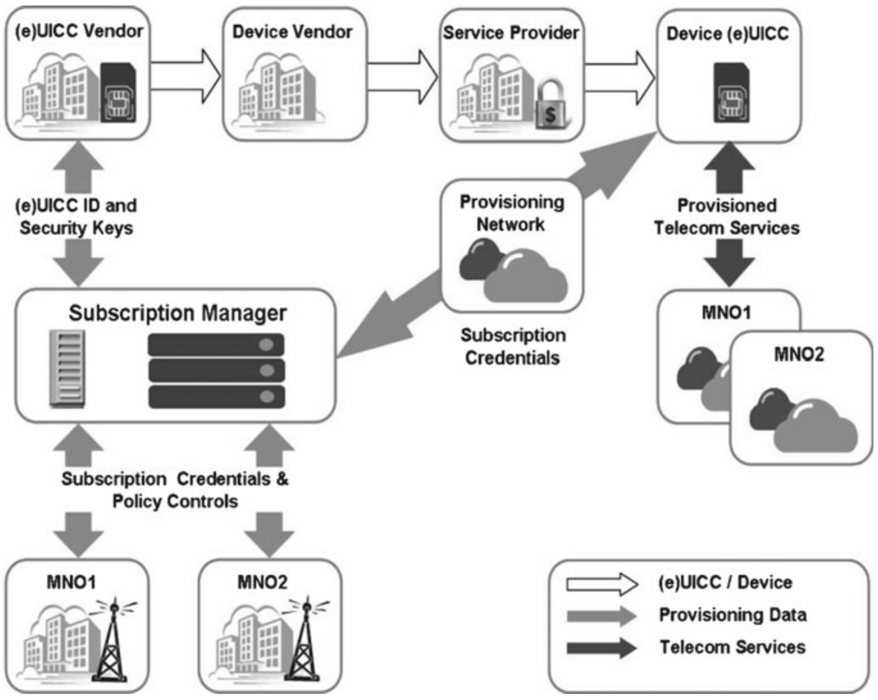


Fig. 4.15 Subscription management ecosystem

would have. Technically there is no great barrier to this solution, but rather its success depend on business issues, standardisation and establishment of the necessary trust relations; the difficulty of the latter should not be underestimated.

The success of this or a similar post-personalisation solution could have a major impact on the future of mobile bearer security and indeed the role of the network operator. The impact might not just arise from the growth in M2M solutions, but from the process being used for all mobile phones. The argument being that if the process and resulting security is suitable for long-lived M2M solutions then why not for shorter-lived mobile phones? This could herald the eventual demise of the SIM in its removable card form, but as long as the eUICC has appropriate defensive capabilities this is more of a business shake-up rather than a security issue. If the future unfolds this way, then there will be temptation to absorb the eUICC into the phone processor, which would be more of a concern. The security protection of the phone processor will no doubt improve over time, however the SIM's success has largely been down to its simplicity, standardisation and ease of personalisation and replacement. Phones are ever more complex, and proprietary solutions abound that are not exposed to public scrutiny and expert peer-review; they are extremely attractive attack targets and a bricked phone is much more of a calamity for a user than a damaged SIM.

4.12 Conclusion

Mobile communications has provided one of the great success stories for smart cards both in terms of the sheer numbers used, the rigorous standardisation and the technical sophistication. Authentication, confidentiality and security in general have always been at the heart of the GSM SIM and the enhancements made for the 3G USIM have resulted in an even stronger and more flexible security solution. Whereas the SIM was historically a single application card, the UICC platform concept now allows the SIM, USIM and indeed many other applications to flexibly co-exist on the same card. The SIM has always represented a value added service enabler and there are some very powerful and standardised SIM Toolkit commands. These facilities have been used to great advantage but never to their full potential, as handset support for SIM Toolkit has always lagged behind the standards. The handset developers are perhaps more interested in ways of exploiting SIM functionality within their mobile applications, e.g. via JSR177 or other APIs. Exactly how the SIM will interact with the handset and indeed other entities in the long-term future is quite difficult to predict as some radical leaps forward in SIM functionality have been proposed, but largely ignored by manufacturers. Adding NFC, a high-speed interface, server-on-card and high capacity storage, challenge many of the past limitations of SIMs, but only when they are consistently implemented and deployed. It is worth noting that despite its success, the SIM is not very card-like and might be better described as a removable security token. If we find from our M2M solutions that we can securely configure and personalise security chips embedded in our smart phones then it does suggest that the days of the SIM card (but not necessarily the chip) may be numbered. This could herald a seismic shift in the power of network operators, phone manufacturers, OS and service providers. There are also phone processors that offer proprietary security solutions and although they currently fall short of the attack resistance and security assurance provided by conventional smart cards and chips, they may increasingly be accessible to application and service developers, whereas SIM cards are not.

Whether the SIM will look the same in future, whether it merges into some kind of SE, and whether it will remain removable is perhaps less important than ensuring than the tamper-resistant security functionality and service enabling features continue to underpin the integrity and usefulness of mobile communications. Any changes that might reduce the current levels of security on a real-time essential solution that serves billions of users, may pave the way for future disaster.

References

1. 3GPP, TR 35.935: Universal Mobile Telecommunications System (UMTS); LTE; Performance evaluation of the Tuak algorithm set, version 12.0.0 Release (2015)
2. 3GPP TS 35.206: 3G Security; Specification of the MILENAGE algorithm set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*; Document 2: Algorithm specification (2014)

3. 3GPP, TS 35.231: 3G Security; Specification of the TUAK algorithm set: A second example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*; Document 1: Algorithm specification (2014)
4. 3GPP2, Removable User Identity Module (RUIM) for Spread Spectrum Systems, 3GPP2 CS0023-C V2.0 October 2008
5. G. Bertoni, J. Daemen, M. Peeters, and G. van Aasche, The KECCAK Reference, version 3.0, 14 (2011)
6. CC, Common criteria for information technology security evaluation part1: Introduction and general model, version 3.1 release 4, (2012)
7. Comprion, IT3 Move 2 Datasheet, http://www.comprion.com/en/products/monitoring/move_2/overview, cited 25 Apr 2016
8. Damien Giry (2015), *Cryptographic Key Length Recommendations,* Keylength.com, http://www.keylength.com/en/3/, cited 25 Apr 2016
9. David Wagner and Ian Goldberg (1998), GSM Cloning, ISAAC Berkley, http://www.isaac.cs.berkeley.edu/isaac/gsm.html, cited 25 Apr 2016
10. ETSI SCP Group, *SCP Specifications*, https://portal.etsi.org/tb.aspx?tbid=534&SubTB=534,639,640,714, cited 25 Apr 2016
11. ETSI, TS 102 221, Smart Cards; UICC-Terminal interface; Physical and logical characteristics, Release 8 (2009)
12. European Technical Standards Institute (ETSI), http://www.etsi.org/, cited 25 Apr 2016
13. Garg V.K, *IS-95 CDMA and cdma 2000*, Prentice Hall 2000
14. GSM Association, http://www.gsma.com/, cited 25 Apr 2016
15. F. Hillebrand, GSM & UMTS - *The Creation of Global Mobile Communication*, Wiley 2002
16. International Organization for Standardisation, ISO 7816 Parts 1-4, http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=54089, cited 25 Apr 2016
17. International Standards Organisation, http://www.iso.org/, cited 25 Apr 2016
18. Java Community Process (JCP), http://jcp.org/en/home/index, cited 25 Apr 2016
19. Java Community Process, JSR177, https://jcp.org/en/jsr/detail?id=177, cited 25 Apr 2016
20. M. Mouly, M-B Pautet, *The GSM System for Mobile Communications,* Cell & Sys. Correspondence 1992
21. Mayes K and Markantonakis K On the potential of high density smart cards, Elsevier Information Security Technical Report Vol 11 No 3 2006
22. K. Mayes, S. Babbage, and A. Maximov, Performance Evaluation of the new TUAK Mobile Authentication Algorithm, ICONS16/EMBEDDED2016 p 38–44 (2016)
23. Near Field Communication (NFC) Forum http://www.nfc-forum.org/, cited 25 Apr 2016
24. NIST, Announcing Draft Federal Information Processing Standard (FIPS) 202, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, and Draft Revision of the Applicability Clause of FIPS 180-4, Secure Hash Standard, and Request for Comments, (2004)
25. Open Mobile Alliance OMA, http://www.openmobilealliance.org/, cited 25 Apr 2016
26. Security Algorithms Group of Experts (SAGE), https://portal.etsi.org/TBSiteMap/Sage/ActivityReport, cited 25 Apr 2016
27. The Javacard Forum https://javacardforum.com/, cited 25 Apr 2016
28. Third Generation Partnership project (3GPP), http://www.3gpp.org/, cited 25 Apr 2016
29. Third Generation Partnership Project, Digital cellular telecommunications system (Phase 2+); *Subscriber Identity Modules (SIM);Functional characteristics* (GSM 02.17 version 8.0.0 Release 1999)
30. Third Generation Partnership Project, *Security mechanisms for the SIM application toolkit; Stage 2* (Release 1999) TS 03.48 V8.9.0 Jun 2005
31. Third Generation Partnership Project, *Security related network functions* (Release 1999) TS 03.20 V8.6.0 Dec 2007
32. Third Generation Partnership Project, *Specification of the Subscriber Identity Module-Mobile Equipment (SIM - ME) interface* (Release 1999) TS 11.11 V8.14.0 Jun 2007
33. Third Generation Partnership Project, *Specification of the SIM Application Toolkit for the Subscriber Identity Module - Mobile Equipment (SIM - ME) interface* (Release 1999) 3GPP TS 11.14 V8.18.0 Jun 2007

34. Third Generation Partnership Project, Technical Specification Group Core Network and Terminals, USIM and IC card requirements, (Release 13) TS 21.111 V13.0.0 Jan 2016
35. Third Generation Partnership Project, Technical Specification Group Core Network and Terminals, UICC-terminal interface; Physical and logical characteristics, (Release 13) TS 31.101 V13.1.0 Jan 2016
36. Third Generation Partnership Project, Technical Specification Group Core Network and Terminals, Characteristics of the Universal Subscriber Identity Module (USIM) application, (Release 13) TS 31.102 V13.3.0 Mar 2016
37. Third Generation Partnership Project, Technical Specification Group Core Network and Terminals, Universal Subscriber Identity Module (USIM) Application Toolkit (USAT) (Release 13) TS 31.111 V13.3.0 Mar 2016
38. Third Generation Partnership Project, Technical Specification Group Core Network and Terminals, Security mechanisms for the SIM application toolkit; Stage 2, (Release 5) TS 23.048 V5.9.0 Jun 2005
39. Third Generation Partnership Project, Technical Specification Group Core Network and Terminals, 3G Security; Security architecture, (Release 13) TS 33.102 V13.0.0 Jan 2016
40. Third Generation Partnership Project 2 (3GPP2), http://www.3gpp2.org/, cited 25 Apr 2016
41. Universal Serial Bus (USB) Forum, http://www.usb.org/, cited 25 Apr 2016
42. Vodafone, SIM/USIM Cards Applications & Security for Mobile Telephony, Masters Lecture Slides at Royal Holloway University of London 2006

Chapter 5
Smart Cards for Banking and Finance

Konstantinos Markantonakis and David Main

Abstract The banking industry managed global payment for consumers over several decades using magnetic stripe card technology. Fraud has always been an issue to manage and it grows over time with the increase of card usage. This coupled with fresh technologies on the horizon led to the adoption of smart card technology to bear down on fraud and open the way for new forms of payment. The major Payment System operators developed a global specification (EMV) describing the technical and security requirements for the physical card to terminal environment, whilst other initiatives addressed contactless opportunities and Internet Payment. This chapter looks into the technical and security details of the EMV specifications along with aspects of the additional security in e-commerce and enhanced smart card-based authentication.

Keywords Payment cards · Magnetic stripe cards · EMV · Chip · PIN · Dynamic passcode · CNP · 3D secure · E-commerce · Token authentication

5.1 Introduction

Both of the ancient civilisations of Greece and Rome developed banking structures and services that accepted deposits, offered loans and provided currencies that could be exchanged for gold or other valuables. These services allowed transactions to take place between businesses and merchants located in different cities with detailed records kept for financial transactions, deposits and interest charges, which allowed disputes to be reconciled and taxes to be fairly levied. Indeed, it can be argued that

K. Markantonakis
Smart Card Centre, Information Security Group,
Royal Holloway, University of London,
Egham, Surrey TW20 0EX, UK
e-mail: k.markantonakis@rhul.ac.uk

D. Main (✉)
DMTechno Ltd., Reading, UK
e-mail: dave@dmtechno.co.uk

K. Mayes and K. Markantonakis (eds.), *Smart Cards, Tokens, Security and Applications*, DOI 10.1007/978-3-319-50500-8_5

without these services to underpin society and imperial adventures, neither civilisation would have developed to the extent that it did.

In the intervening centuries, Payment Systems have evolved significantly, becoming regulated and playing an important role in the development of the interconnected global trading economies that are now central to the interests of nearly all individuals, organisations and nations. It was around the seventeenth century that paper-based currency was introduced, breaking the connection between a token and its actual worth. This was closely followed by other financial developments including cheques, money-orders and other ways to move money between accounts.

During the 1920s and 1930s a number of schemes were developed to allow customers to pay for goods and services without needing cash or a cheque, with examples to be found in the telegraph and fuel industries. To some extent these were extensions of the traditional services offered by some grocers, butchers and bakers that delivered goods on request with the household settling the bill at the end of the month. It was not until the 1950s that the banking industry began to get interested and started to experiment with larger scale card based schemes that allowed any cardholder to shop in a store that accepted the cards, without them having a prior relationship. A group of banks on the west coast of the USA developed BankAmericard, which evolved into Visa and a group on the east coast developed a similar scheme which evolved into MasterCard. American Express grew out of Wells Fargo in the mid-west and Diners Club is considered to have been ahead of them all when Frank McNamara came up with a card-based dining scheme, after having embarrassingly been short of cash after a business dinner in a New York steak house [1].

The common denominator is that they all offer cards that can be used in restaurants, hotels and stores as an alternative to cash, with Cardholders usually billed monthly for any purchases. Some offer revolving credit and charge interest if the balance is not paid in full and some have an annual fee. The transactions are posted to bank held accounts via networks—originally paper based but data read from magnetic-stripes on the cards has been submitted electronically since the late 1970s, moving to chip read data around the turn of the past century. Over time, the schemes have expanded from cities, to countries, to the present day global acceptance of most cards. In terms of numbers, it is estimated that currently there are nearly 2 billion cards used in some 30 million Merchant locations [2] and credit accounts are now surpassed by debit products that take funds from a current account as and when a transaction arrives.

This book chapter is organised into the following subsections. Section 5.2 provides an introduction to the concept of payment cards, including magnetic stripe cards. Section 5.3 introduces smart card technology and the EMV standards. Sections 5.4–5.8 introduce various issues around electronic commerce and, in particular, 3-D Secure and Token Authentication. Finally, some concluding remarks are presented in Sect. 5.9.

5.2 Payment Card Technologies

A typical card payment scheme involves four entities; the Cardholder, the Merchant, the Acquiring bank (merchants' bank) and Issuing bank (card Issuer). A typical Payment Card infrastructure is presented in Fig. 5.1.

Often described as "the four corner model", the construction is that the Issuers provide cards to their Cardholders, backed by accounts held by the issuing banks, and on the other side Acquirers sign up Merchants to accept cards for payments. The Payment System establishes a set of rules to administer the system and operates the network which provides global connectivity to route transactions between the issuing and acquiring institutions. When a Cardholder wishes to buy goods or services at a Merchant, they present their card and the Merchant will accept this for payment subject to obtaining an authorisation from the Issuer. This is frequently completed online, but in some circumstances (e.g. low value) the card acts offline to authorise the transaction on behalf of the Issuer. When approved the Merchant will provide the goods or services and later submit a clearing and settlement transaction to their Acquirer. This will be routed by the Payment System to the corresponding issuing bank which will add the transaction to the Cardholder's account and the Merchant will receive the necessary funds. The nature of the account corresponds to the card classification. For credit cards (which is where card payments began), the transactions will be accumulated until the end of the month, when the Issuer will send a statement to the Cardholder for the overall sum. Cardholders have the choice to pay in full, or may choose to only pay part of the bill, in which case they are taking credit and the Issuer will charge interest, billing it to the next statement. Charge cards are similar except that the full amount has to be paid each month. Debit cards are linked to a current account and make a deduction as and when transactions are received. The four corner model is typical of the largest Payment Systems—MasterCard and Visa. In some other schemes, such as American Express and Discover, the roles of the Issuer, Acquirer and network are merged and the Payment System both issues cards and signs up Merchants.

One feature of the early payment cards, which still remains operational today, is that they contained embossed information (e.g. card number, name and expiry date) on the front of the card. A Cardholder that wanted to pay for goods or services would

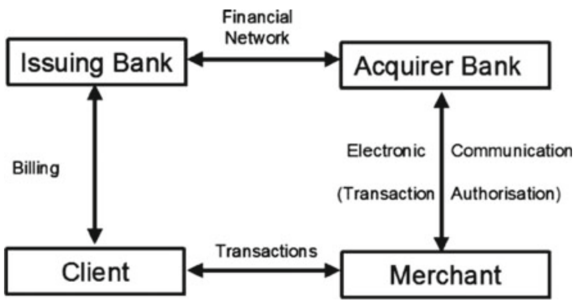


Fig. 5.1 Typical credit/debit card infrastructure

hand their card to the retail assistant and they would use a so-called "zip-zap" swipe machine to obtain a paper imprint of the card information on a multi-sheet payment slip. The retail assistant would complete this with the amount and date and have the Cardholder sign the slip for authentication, checking the signature against that on the signature panel on the back of the card. In some circumstances, the Merchant would make a phone call to the Payment System authorisation centre to obtain approval. The Cardholder would be given one sheet of the payment slip as a receipt and the Merchant would retain the rest for their records and to supply a copy to their acquiring bank. This would then be physically transferred to the Payment System clearing and settlement centre, where it would be processed and forwarded to the issuing bank according to the card number. The Merchant would receive payment and the Issuer would apply the transaction to the Cardholder's account. Fraud control was by means of the calls to the authorisation centres and manual checking by the Merchant of the Cardholder signatures.

Although the system worked well, as the number of cards and transactions grew the logistics of physically transferring the paper slips and controlling the fraud became ever more difficult and some form of automation was needed.

5.2.1 Magnetic Stripe Cards

This came in the form of a magnetic stripe (mag-stripe) added to the back of the card which carried the card information in an electronically readable form. By swiping the card through a slot on the terminal containing a magnetic track read head the card information could be instantly captured and with the amount entered on a keypad by the Merchant and the date and time from an electronic clock within the terminal, all the information for a transaction was available. These transactions could be stored up in a terminal and submitted at the end of the day as a batch to the acquiring bank for clearing and settlement—altogether much more efficient than transferring paper slips. In turn the acquiring and issuing banks were networked to the Payment System clearing and settlement centres allowing the transactions to flow electronically from Merchant to Issuer.

In most cases the terminals connected up using a dial-up modem and traditional telephone lines and the same connection could be used for real time authorisation of transactions. If the transaction amount exceeded the Merchant floor limit (or other risk management checks), before completing the sale the terminal would send the transaction information to the Acquirer as an authorisation message, to be routed via the Payment System to the appropriate Issuer authorisation system, which could in real time check the account for available funds and lost or stolen status. An approval or decline response would be returned via the same route and the terminal would indicate the disposition to the Merchant, the whole process taking a matter of a few seconds.

At around the same time, these systems allowed the introduction of Automated Dispensing Machines (ATMs) for Cardholders to withdraw cash from their current accounts using debit cards, or to take cash advances on credit products. This included the ability to withdraw foreign currency from overseas ATMs and it is interesting to note that this coincided with the lifting of currency exchange restrictions in many countries.

For Merchants, Cardholder authentication was still by signature, manually checked against the panel on the back of the card, but an automated solution was required for the ATMs. This was in the form of the Personal Identification Number (PIN) that could be entered on a keypad on the ATM and submitted in authorisation request to the Issuer for online checking. PIN numbers (typically 4 digits) and the mag-stripe data on the card are an early example of two factor authentication—something the Cardholder owned (i.e. a valid card) and something they knew (i.e. the PIN). In some countries online debit card schemes introduced PIN pads at Merchants to allow Cardholder entered PINs to replace signature checking as a means of cardholder authentication. In all cases of online PIN usage, the PIN data is encrypted for transmission in the authorisation messages.

5.2.1.1 Magnetic Stripe Card Security

Whilst the mag-stripes and electronic terminals greatly increased the opportunities for Issuers to control fraud, protection was required to prevent the creation of counterfeit cards by the construction of a valid mag-stripe from card information gathered from paper receipts. This was in the form of a cryptographically generated check value (Card Verification Value—CVV) included in the mag-stripe data. When the mag-stripe data is received by the Issuer in an authorisation message, they can validate that the check value is correct and detect counterfeit cards. It should be noted that this does prevent the whole strip being copied to produce a cloned card.

The Issuer calculates the check value by taking the card number and expiry date and encrypting them under a symmetric key generated for the purpose. A proportion of the resulting cryptogram is used (via a conversion function) to produce a 3-digit check value that is included in the mag-stripe data. When the track data is received for authorisation the Issuer system will take the received card number and expiry date and recalculate the check value using the same key. If they match then the Issuer has confidence that the data comes from a card that they originally issued. The code is only three digits long due to restrictions on the number of characters that can be encoded on a mag-stripe. The result is that a fraudster could guess the value, but with a one in a thousand chance of success this had not been a problem in practice.

A later development is to offer protection for "Card Not Present" (CNP) transactions, such as e-commerce and telephone order. This is by means of another 3-digit check value, calculated in the same way as above (but with a different key) that instead of being included in the mag-stripe is stamped into the signature panel on the back of the card. This is the Card Verification Value 2 (CVV2), which is asked for by the Merchant and forwarded in the authorisation request. If the value checks out, it indicates that the card details have not just been stolen from a web-server, or captured by a wedge device, but that the purported Cardholder has the physical card in their possession, or at least that capture of card details for fraudulent purposes has to be sufficiently sophisticated to also capture the number on the back of the card. This mitigates large-scale hacking attacks against databases and tends to force fraudsters to target individual cards, thus reducing the impact of a successful attack.

For encryption of the PIN, if the same key is used for all PIN transmissions and the messages are intercepted, it will then be possible to launch a dictionary attack that will match the encrypted PIN messages to unencrypted PINs. This is a well-known issue within cryptography and it can be avoided by adding some extra information to the message that will be encrypted. For the payments industry the solution is that the PIN is XORed with the unique Primary Account Number (PAN) with padding added to bring the message to the appropriate length before it is encrypted. This ensures that different cards that have the same PIN will result in different encrypted PIN messages.

The encryption uses a symmetric encryption algorithm which means that the Issuer needs to decrypt with the same key used to encrypt the PIN block. However, in a global network it would not be possible for an Issuer to know the keys for all the PIN pads on the planet, and thus the Payment Systems offer PIN translation services. In the PIN pad, the PIN block is encrypted using a key unique to the device and known to the Acquirer which handles the authorisation messages. Upon receiving a message, the Acquirer decrypts the PIN block and then re-encrypts it with an Acquirer Working Key (AWK), which is shared between the Acquirer and the Payment System (e.g. Visa or MasterCard). The new encrypted PIN block is then included in the message forwarded to the Payment System, which upon reception, decrypts the PIN block using the AWK shared with the Acquirer. Once the destination Issuer has been determined from the card number, the PIN block is encrypted again using an Issuer Working Key (IWK) shared between the Payment System and the Issuer. The encrypted message is forwarded to the Issuer, who decrypts it and can then

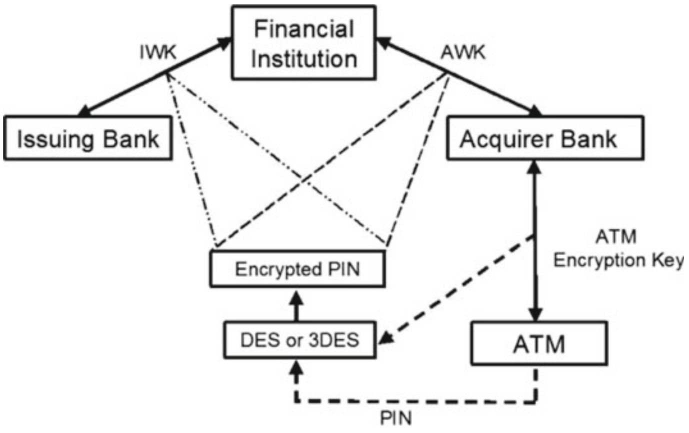


Fig. 5.2 PIN encryption and cryptographic key relationship

check the PIN as part of its authorisation processing. The Issuers and Acquirers have Zone Control Master Keys (ZCMKs) which are shared with the Payment Systems and used to regularly update the AWKs and IWKs. The ZCMKs are also updated, but on a less frequent basis. The above procedure is summarised in Fig. 5.2.

5.3 Smart Cards and EMV

By the early 1990s, it became evident that fraud due to the cloning of mag-stripes was increasing and that controlling the problem through online working and greater system intelligence was becoming less effective. There was also a trend towards more low value purchases, indicating that a solution offering offline authorisation would be beneficial to save on the online costs, plus some markets had a desire to move away from signatures to PIN-based solutions, partly due to experience with PIN-based debit card schemes and also to allow more unattended automated sales in ticket and vending machines.

Of the available technologies, smart cards offered a tamper- resistant platform with strong cryptographic card authentication, the ability to bring enough intelligence to act as the Issuer for offline authorisation, plus the opportunity for PINs to be validated locally in the card. They were thus a natural solution for the replacement of mag-stripes. It could also be seen that the same basic technology of a chip embedded in a plastic card was being developed for contactless ticketing applications and that this could be of future interest for payments.

To ensure that a single terminal would be able to read and interact with smart cards issued through any of the Payment Systems, it was apparent that an industry specification would be required and the major Payment Systems agreed to work together to this end. The result in 1996 was the EMV specifications which describe the interoperable requirements for the card to terminal interface, from the low level electrical parameters through to the functional exchange of data and cryptographic processes. The EMV name stems from the three Payment Systems at the time: Europay, Mastercard and Visa, [3] but with changes to the industry players and the creation of EMVCo as a management body with a wider scope than just smart cards, EMV is now a payment industry term covering a whole range of specifications, testing and management.

EMV draws its baseline security from the underlying security functionality and tamper-resistance of the microprocessor chip in the smart card. This is capable of securely storing cryptographic keys and other sensitive information such as reference PINs. EMVCo runs an evaluation programme to certify that chips reach satisfactory security levels and are resistant to attacks such as power analysis and glitching.

The EMV specification describes the following security functions:

1. Offline Data Authentication
2. Card Authentication
3. Issuer Authentication

4. Transaction Certificate
5. Offline PIN Verification
6. Management Functions

The main EMV processes, relating to authentication, are described in the following sections.

5.3.1 Card Authentication (Offline Data Authentication)

Authentication of the data from the card indicates to the terminal that the data and public key in the card have not been tampered with and remain exactly as placed on the card by the Issuer during personalisation. There are three card authentication methods defined in the EMV specifications and all three methods are based on the existence of a Public Key Infrastructure (PKI) established by the relevant Payment System. This is at three levels and uses a Payment System Certification Authority (CA) to sign public keys for Issuers which in turn are used to sign data and public keys on the cards.

The Payment System holds the root private key in its CA and distributes the corresponding public key into every EMV terminal. Issuers generate their own key pairs and send the public key to the CA to obtain a key certificate. They in turn use their private key to produce a key certificate for the card public key that is loaded on the card, together with a copy of the Issuer certificate.

The three methods specified in the EMV specification are described below:

5.3.1.1 Static Data Authentication (SDA)

This method was introduced when cards had no public key cryptographic capability and has since been deprecated as card technology has evolved. It is described here for completeness. During card personalisation, the Issuer signs certain card-specific data using its private key and places the resulting signature in the card, together with a copy of its own public key certificate. When the card is presented to a terminal and SDA is selected, the terminal retrieves the Issuer's public key certificate and validates it using the payment system public key present in all terminals. During this process the Issuer public key is extracted and the terminal knows that it is genuine. The next step is to retrieve the signed data from the card and validate it using the Issuer public key. If the process is completed without any problems, the terminal obtains the necessary reassurance that the card data has not been modified since the card was personalised. All of the public key cryptography and keys use RSA as the algorithm. This process is described in Fig. 5.3.

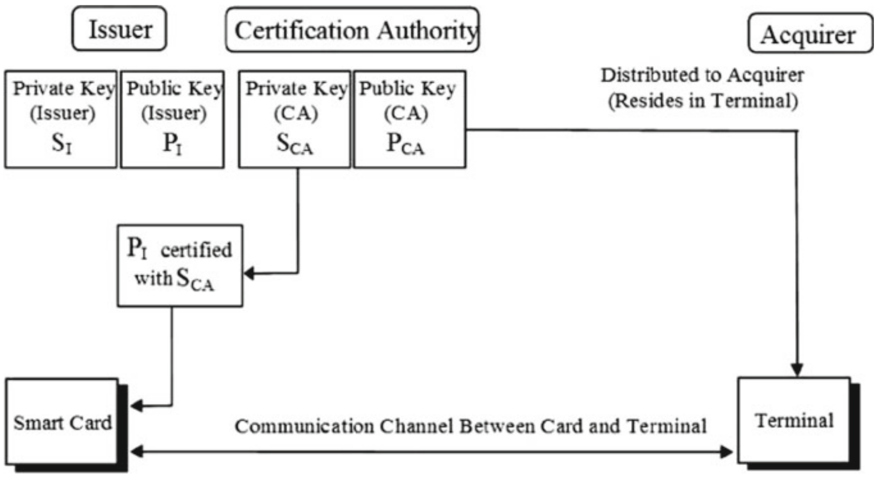

Issuer
Certification Authority
Acquirer
Private Key (Issuer) S_I
Public Key (Issuer) P_I
Private Key (CA) S_{CA}
Public Key (CA) P_{CA}
Distributed to Acquirer (Resides in Terminal)
P_I certified with S_{CA}
Smart Card
Communication Channel Between Card and Terminal
Terminal

Fig. 5.3 Static Data Authentication

5.3.1.2 Dynamic Data Authentication (DDA)

This follows a similar process in terms of validation of the Issuer public key and the whole process is summarised in Fig. 5.4. Given a valid Issuer public key the next step is to retrieve the card public key certificate and validate it using the Issuer public key. The terminal then sends a challenge to the card including a random number (for uniqueness), the card signs this together with card specific data using its private key

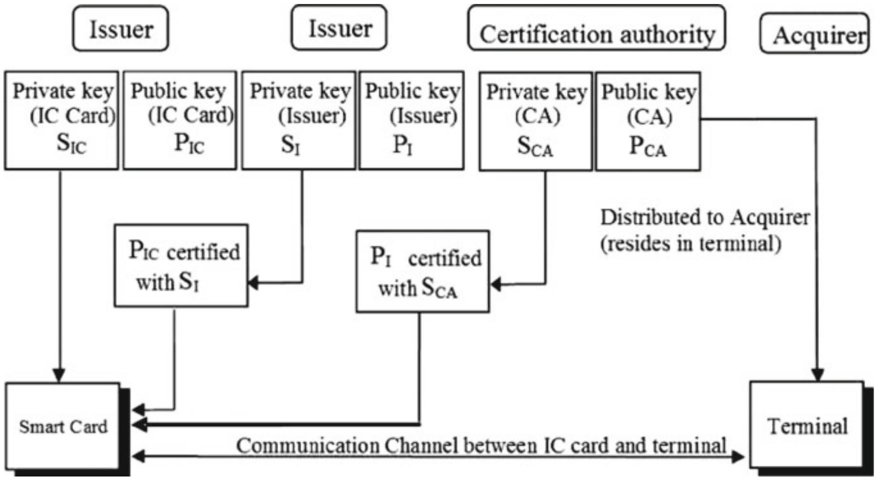

Issuer
Issuer
Certification authority
Acquirer
Private key (IC Card) S_{IC}
Public key (IC Card) P_{IC}
Private key (Issuer) S_I
Public key (Issuer) P_I
Private key (CA) S_{CA}
Public key (CA) P_{CA}
Distributed to Acquirer (resides in terminal)
P_{IC} certified with S_I
P_I certified with S_{CA}
Smart Card
Communication Channel between IC card and terminal
Terminal

Fig. 5.4 Dynamic Data Authentication

and the terminal verifies the resulting signature. If the process is completed without any problems the card has been dynamically authenticated and the data shown to be unaltered.

5.3.1.3 Combined Data Authentication (CDA)

A further enhancement in data authentication is to include the application cryptogram (see next section) in the data signed by the card. This ensures that wedge devices are unable to modify the cryptogram used for online card authentication.

5.3.1.4 Card/Issuer Authentication

Card authentication is an online process that takes place between the card and its Issuer and uses symmetric cryptography. If the terminal has reached the position that it knows the transaction will be sent online for authorisation, it requests an Application Request Cryptogram (ARQC) from the card. Along with the request, the terminal sends a number of data elements relating to the transaction, including a random number to ensure freshness and defend against replay attacks.

Using a 2TDES (2 key triple DES) key loaded during personalisation, the card takes the data supplied by the terminal and encrypts it along with internal data from the card to generate a cryptogram. One of the internal data elements is the Application Transaction Counter (ATC), which is incremented for each transaction and consequently ensures uniqueness for the cryptogram output.

A typical set of data to be input to the cryptogram is shown in Table 5.1.

Table 5.1 Typical values supported within the ARQC

Value	Source
Amount, authorised (numeric)	Terminal
Amount other (numeric)	Terminal
Terminal country code	Terminal
Terminal verification results	Terminal
Transaction currency code	Terminal
Transaction date	Terminal
Transaction type	Terminal
Unpredictable number	Terminal
Application interchange profile	Smart card
Application transaction counter (ATC)	Smart card

The cryptogram (ARQC) is included in the authorisation request to the Issuer together with all the card and transaction data that was input for the card calculation. As part of the authorisation processing, the Issuer repeats the cryptogram calculation using the same data and its copy of the key personalised into the card. If all is in order the cryptogram output will match that sent by the card and the Issuer has confirmation that the card is genuine and that the transaction data has not been modified. This result is included in the various authorisation checks, (including funds availability, lost, stolen, etc.) with the approve/decline result returned to the terminal in the authorisation response message.

It may be noted that normally card keys are derived from a master key (based on card number) and that the actual card key used will be a session key. Thus, the process to check the cryptogram needs to first derive the card key from the card number and then calculate the session key from the ATC.

The value of the ATC is also of interest as a repeated value could indicate a transaction replay and a sudden increase in value could indicate that the card has been subject to attack.

Issuer Authentication: an Authorisation Response Cryptogram (ARPC) may be sent in the authorisation response from the Issuer to be forwarded by the terminal to the card. The ARPC is a 2TDES calculation on the ARQC using the same key used for the ARQC processing.

When the card receives the ARPC, it verifies it against the ARQC calculated previously using the same key. If all is in order the card has confirmation that it has been in communication with its own Issuer and will perform pre-programmed activities, such as the resetting of the offline usage counter. Since the ARPC is linked to the ARQC and the ATC, any attempts to manipulate the card by replaying previously captured ARPCs will fail.

5.3.1.5 Transaction Certificate

The Transaction Certificate (TC) is also a cryptogram, calculated in a similar manner to the ARQC, including data that indicates what form of offline data authentication was used, whether the PIN was checked and the outcome of the transaction. It therefore forms a cryptographically protected "receipt" that can be used to resolve disputes.

5.3.1.6 Offline PIN Check

Offline PIN verification consists of checking a PIN entered by the Cardholder against a reference PIN loaded into the card by the Issuer. The result is indicated to the terminal and included in data from the card input to the ARQC. Retries are limited and controlled by a counter. PINs may either be transmitted as cleartext—appropriate for when the PIN pad and card reader are an integrated tamper resistant unit—or encrypted under a public key from the card.

Management Functions: in a similar manner to the ARPC, the Issuer may also prepare Issuer Scripts that are forwarded to the card by the terminal. These consist of commands to the card protected by a secure messaging MAC and when necessary encryption. Separate keys to that for the ARQC/ARPC are used. These commands might be used to update parameters, such as those that control offline usage, clear the PIN try counter for a blocked card, or to change the PIN itself. In the latter case the payload (PIN) is encrypted.

5.3.1.7 EMV in Practice

Migrating from mag-stripe cards to smart card technology has been an evolutionary process. The early cards did not support active public key cryptography and had limited non-volatile (Electrically Erasable Programmable Read-Only Memory (EEPROM)) capacity. This resulted in the cards being restricted to using SDA, and the Payment System public keys being short at 512 bits in order that the Issuer certificates would fit in the cards. The Payment Systems offered guidelines and advice to support the participating banks and in some regions chip incentive plans were introduced to initiate the rollout. Network support for chip data and terminal performance for the public key calculations were also issues that needed to be addressed.

Among the forerunners for the adoption of smart card technology was France, which had a domestic smart card scheme running in advance of the EMV introduction. The UK was the first country to roll-out EMV, starting with "Chip & Signature", but moving to "Chip & PIN" once cardholder confidence was established. Other European countries soon followed suit, with France migrating from their domestic programme. Major countries in Asia Pacific and South America also adopted EMV, with Canada in North America. The USA has been slow to move to chip, partly because of the efficiency and low costs of its networks and it is only now (2015) that adoption is beginning.

One of the driving forces behind the adoption of chip is the liability shifts introduced by the Payment Systems and national payment bodies. For example, the 2005 liability shift in the UK stated that, "in case something goes wrong in a transaction, the cost will be taken by the party (i.e. Issuer, Acquirer or Merchant) that does not have the capability to process an EMV transaction". As technology improvements and economies of scale increased, the cost/performance of both cards and terminals it became possible to adopt the data authentication methods using active public key cryptography. This was accelerated by the development of contactless products targeted at fast (no PIN) low value transactions, which needed offline authorisation and consequently public key capable cards.

Also improvements in factorisation techniques meant that 512-bit RSA keys would no longer be safe and key lengths have gradually increased following the planned path for replacement with longer keys to maintain the security strength. The current 1408-bit and 1984-bit key lengths are expected to serve for the foreseeable future. The 1984-bit length is less than the common 2048-bit at this strength, due to the low level transportation protocol in EMV that has a maximum block size limit.

Where chip has been implemented fully, the amount of fraud from counterfeit cards and lost and stolen cards has fallen to very low levels. The main problems have been with migration of fraud into other payment areas, in particular the remaining mag-stripe environments and e-commerce. Having PIN usage at a large number of Merchants, compared to just a limited number of ATMs, has increased the opportunities for fraudsters to capture card details with matching PIN. Whilst this information is of little value in a chip acceptance environment, it can still be useful in the old mag-stripe acceptance environment, especially ATMs, and some fraud has been seen along these lines. Increased fraud in the e-commerce environment has largely been due to the explosion of Internet purchases—as a proportion of transaction volume it has changed little—the issue really lies with Merchants who are willing to take payments with only minimal security checks.

The existing EMV specification is now two decades old and the RSA key sizes needed to maintain security will eventually become unwieldy. Migration to a new system will take some 15 years using natural replacement cycles and thus a new version of EMV is under development. This will take advantage of the increased capability of the cards, which can now make the Issuer risk management decisions internally, rather than supplying data for the terminal to process on the Issuer's behalf. It will use ECC (Elliptic Curve Cryptography) and feature a secure channel between the terminal and card, preventing wedge and shim attacks and defending against relay attacks, in addition to addressing Privacy concerns relating to contactless eavesdropping.

5.4 Cardholder Not Present Transactions

If we cast our minds back just a few years ago then our perception of banks, banking and financial transactions was quite different to what it is today. Banks were in the high street, in real buildings, with real people and generally inspired some confidence. It is doubtful that anyone stepping through the front door would have questioned whether they were entering a genuine branch, or that deposited funds were kept in any other form than cash or bullion locked in an impressive looking vault. Similarly, when spending our hard-earned cash, we would wander into a real shop and walk away with a bag of physical items. For the customer, the distinction and level of risk between buying from a brand name store and a shady looking market trader was clear to see. Similarly, the Merchant had some clues to the authenticity of the customer and their presented credentials, because both the Cardholder and the card were physically present. Fast forward to today and there is a much greater proportion of transactions where the "Card is Not Present" (CNP). One immediately thinks of the Internet as the driving force for this, but we have in fact being generating CNP transactions for a long time by purchasing items over the telephone. One might expect that in an age of smart cards and ever smarter technology, that the CNP problem is reducing, but alas, technology has made CNP transactions an even more attractive target for criminals. One reason for this is that EMV [4] Chip & PIN has tightened the security around normal cardholder present transactions and so criminals are switching to the softer target of CNP for telephone sales and of course Internet purchases. In CNP, not only is the entire chip solution not possible, but so too is the relatively weak electronic security of the mag-stripe and indeed even the customer signature. As a starting point, let us consider some of the ways in which this can be exploited.

5.4.1 Purchase from a Genuine Merchant Using Someone Else's Payment Details

In the first instance let us assume that we have found an old-style purchase receipt, made using the victim's card. The receipt was in the rubbish bin outside the victim's home—so we also know where they live. If we can determine the card number, name and expiry date, we could try and order something over the telephone or via the Internet. One of the initial barriers to this was that purchased goods could only be delivered to the same address as the card was registered to, but this has been relaxed and many valuable items today are not even physical, e.g. software, e-books and music downloads. To improve the situation most receipts no longer show the full card number and a Merchant may challenge the purchaser to state the Card Verification Value (CVV2) [5], a three-digit code stamped into the back of the Payment Card. These measures may make it less attractive to search through dustbins for receipts, however, the measures can be undermined by anyone who has seen the physical card just once. Considering how many times and in how many places, a payment card is presented, the opportunity to make a note of the details, take a second swipe, or just a quick photo with a mobile Phone, is hardly limited.

5.4.2 Genuine Purchaser Buying from a Rogue Merchant

It seems that we are buying more and things online or via the telephone, but how do we know we are dealing with genuine merchants? The transaction has to start with a telephone number or a website address and so the question is how the purchaser obtained it. If it is trusted information (used before) or closely linked to another trusted source (e.g. obtained from your bank) then there is less likely to be a problem—however we are often first-time buyers following on from a directory or Internet search. The contact information may have come from a quality publication and so some confidence can be taken from the publisher's policing of advertisers and indeed the significant expense involved in placing the advertisements. Information from the Internet is less certain and although some sites have certificates that may be checked, very few customers are sufficiently knowledgeable to do this. Whatever safeguards we take and however careful we are selecting merchants, there comes a point when we hand over our details including name, address, card number, expiry date and indeed the CVV2. If we click on the save details box then this information is stored on the merchant's database and if we buy from many merchants the details end up on many databases. With so many merchants in the fast moving competitive world of Internet and telephone sales, it is perhaps naive to expect them all to have perfect processes for the management and protection of stored transaction data. It only takes one rogue Merchant or employee to use/sell our customer details for fraudulent purposes for a problem to develop [6].

5.4.3 Third Party Attacker

A third-party attacker or hacker may seek to discover the transaction information. Whilst eavesdropping on telephone calls or bribing a Merchant's employee might be effective methods, they are resource intensive for the fraudster and rely on an element of luck—we will focus on the more technical possibilities. Internet IP traffic could be grabbed in transit by a hacker, but they would have something of a "needle in a haystack" problem, leading most attackers to concentrate on areas where they are certain to obtain useful information. Hacking into a merchant server which stores card data is one route and has resulted in some high profile breaches, such as Carphone Warehouse [7], Hyatt Hotels [8] and Global Payments [9]. This is bad news for individual customers, but is worse for the merchant, because of the bad publicity, compensation claims and loss of customer confidence. Another rich seam to mine is to persuade customers to send the information direct to the attacker. This may sound like an odd thing for a consumer to do, but that is what "Phishing" [6] is all about. For example, a customer receives a genuine looking email suggesting that they re-submit their payment details to their "bank", as part of a routine check. Of course there is no bank except for one safeguarding the ill-gotten gains of the criminal. Phishing is a major problem and it can be combined with attacks against the customer's Personal Computer (PC). If the customer clicks on an attachment to a fraudulent email, it may install malicious software known as "malware" [10] onto the computer. The malware could capture keystrokes and screen information for reporting to the criminals.

It would seem from the foregoing examples that some paranoia over one's card and transactional details is quite justified for CNP transactions. However the situation that the information that is sneaked, copied or stolen is then used to make counterfeit cards for Cardholder present transactions in countries that do not yet use Chip & PIN is not as bad as might be imagined. This is since the mag-stripe track is not available to be scanned in the CNP world and thus the cryptographic CVV in the track data (see Sect. 5.4.2) is not known. When used for an online transaction, such a cloned card should be recognised and declined by the Issuer.

Clearly CNP transactions fall way short of the security that we are becoming accustomed to with Chip & PIN and alternative strategies are required to improve the situation. Many merchant sites already perform basic checks such as address verification, CVV2 checks and in some cases customer device profiling. For the latter the merchant may log certain information about a customer's device, such as internal numbers, cookies or their ISP and if the same information is present on subsequent transactions they will have confidence that it is not a fraudster. However, these approaches do not actually verify the identity of the Cardholder and to remedy this, the Payment Systems introduced a protocol that allows an issuing bank to authenticate their Cardholder during an e-commerce transaction.

5.4.4 3D-Secure

The 3D-Secure (3D-S) solution has been widely adopted by the Payment Systems under different brand names, such as MasterCard "SecureCode" [11] and "Verified by Visa" [12]. The protocol breaks up the Internet transaction into a number of conceptual domains. There are in fact three domains such that each entity or supporting system sits within one or more domains. The interrelation of the domains is shown graphically in Fig. 5.5.

Within the Issuer Domain, we see that the Issuer has the relationship with the Cardholder. This implies that previously a Cardholder registration process has been completed and so the Issuer is well placed to authenticate the Cardholder remotely using a server located within the Issuer Domain, known as the Access Control Server (ACS). The Acquirer Domain contains the Merchant, acquiring bank and associated network systems. Since most merchant e-commerce server systems do not have the 3D-S protocol built in, a Merchant Server Plug-in (MPI) provides the necessary functionality to initiate the authentication in the Issuer Domain. This leaves the Interoperability Domain, which is basically the "plumbing" used to connect the Issuer and Acquirer Domains together, for the purpose of securing e-commerce transactions. The main component within the Interoperability Domain is the Directory Server (DS), hosted by a Payment System in order to determine which ACS to query when a particular card is involved in a transaction. The directory is effectively a mapping of participating/registered cards to the appropriate ACS server address. Also within

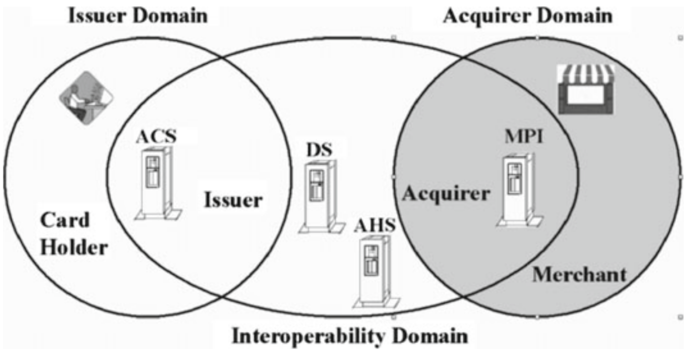


Fig. 5.5 3D-Secure domains

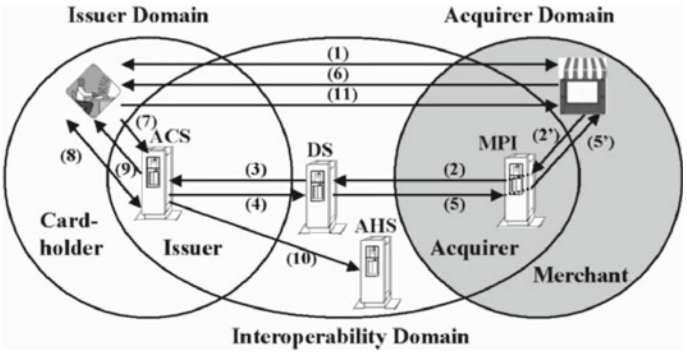


Fig. 5.6 Example message flows in 3D-Secure

this domain is the Authentication History Server (AHS), which maintains a log of authentication history for the purposes of managing disputes. Once authenticated, a transaction will be authorised and cleared through the usual Payment System networks connecting Acquirers to Issuers. The detailed connections between the server systems is beyond the scope of this chapter, but the reader can assume that best practice IT security protocols are in use (e.g. Secure Sockets Layer (SSL) [13]).

The way that all these entities work together and the associated message flows is best illustrated by an example. Figure 5.6 represents the basic protocol steps when a Cardholder buys a product via a merchant's website. Note that the MPI would normally be managed by the Merchant but for the purposes of explanation we will treat it as an Acquirer element that simply acts as a conduit for website generated authentication requests and responses.

The explanation for the sequence of message flows is given in Table 5.2.

5.4.5 *Thoughts on 3D-Secure*

The real test of 3D-Secure is whether customers and Merchants like it. At present Cardholders do net get a lot of choice, because if they need to buy from a participating Merchant they are driven into using the scheme. Perhaps they are allowed one or two transactions when they can opt out, but then the next time they are compelled to register. Having registered, the customer then has to go through the 3D-Secure process at other participating Merchant sites. Even registering is not necessarily a good experience for the Cardholder. In a world where our banks are continually warning us about phishing and keeping our transaction details safe, 3D-Secure pops up an unexpected window asking us to enter sensitive financial information. In use, the 3D-Secure experience depends on the complexity of the authentication challenges that the issuing bank decides is appropriate. For example, the Cardholder may be asked to submit randomly positioned letters from a long pass phrase, which is by no

Table 5.2 Sequence of message flows

Flow number	Description
(1)	This simply represents the Cardholder browsing for products on a merchant's website and beginning the "check-out" for online purchase
(2)	After the Cardholder has entered their normal payment details, such as card type/number etc., the website needs to determine if this card has been enrolled in the 3D-Secure scheme. It does this by generating (via the MPI) a Verify Enrolment Request message (VEReq), which is sent to the DS
(3)	The DS can recognise the issuing bank from the number range of the Payment Card and if the Issuer is registered as provisioning 3D-S to their Cardholders, the DS forwards the request to the appropriate ACS
(4)	The ACS looks up the particular card number in its database and determines whether the Cardholder is registered for the scheme and then returns a Verify Enrolment Response (VERes) message to the DS
(5)	The response is then returned to the Merchant via the MPI and the Merchant's action depends on the status and local policy, e.g.: • Not registered—allow the transaction to proceed anyway • Not registered—suggest that the Cardholder registers but allow the transaction to proceed • Not registered—force the Cardholder to register or abort the transaction • Registered—follow the steps below to complete the transaction
(6)	The Merchant (via the MPI) now sends a Payer Authentication Request (PAReq) message to the cardholder's browser. This essentially opens a new browser window and initiates a "POST" to the appropriate ACS using the URL returned in the VERes
(7)	The Payer Authentication Request is effectively forwarded by the browser in the cardholder's terminal/PC to the appropriate ACS
(8)	The ACS creates the content of the cardholder's browser window, which will include some kind of authentication request such as entry of a password or one-time passcode. The cardholder entered information is returned to the ACS
(9)	The ACS checks the information from the Cardholder and uses it as input to a Payer Authentication Response (PARes) message. The response also contains an Accountholder Authentication Value; the details of which vary depending on the card scheme. MasterCard uses UCAF (Universal Cardholder Authentication Field) and Visa uses CAVV (Cardholder Authentication Verification Value). The response is digitally signed and then returned to the MPI via the cardholder browser and the authentication window is closed. If the Cardholder information is wrong, most systems will indicate this and allow a limited number of re-tries
(10)	The ACS also sends a copy of the PARes to the AHS for use in dispute handling
(11)	When the PARes reaches the Merchant via the MPI, a positive result will normally result in the Merchant proceeding with the transaction as normal, seeking authorisation via the Acquiring bank. The Acquirer will pass on the request to the Issuer who is now more likely to approve the request, as it will obtain the Accountholder Authentication Value obtained via the 3D-Secure process

means trivial unless the phrase has been written down. Whilst there are good security reasons for all of this, it seems to be heading in the opposite direction to merchant strategies, which would really like us to buy something instantly on an impulse click of the mouse button. It also seems to conflict with the financial industry's drive toward new wave/touch and pay methods of purchasing. In recognition of this, work is underway on a revision of 3D-Secure that will use device information to allow Issuers to recognise Cardholders through their connecting devices and make the process for lower risk transactions transparent to the Cardholder, who would not be asked to actively authenticate.

5.5 Dynamic Passcode Authentication

There are many authentication and transactional systems that make use of static codes, for example, a log-on password could be regarded as a static code. The major problem with this is that once discovered, a static code no longer offers security. Furthermore, if the code has to be memorised by a user it tends to limit its length and complexity and most likely a dictionary attack [14] would identify the code in far fewer attempts than the code length would theoretically imply. One approach is to make the code dynamic so that once discovered it quickly loses its value; however, humans are not great at remembering codes, especially if they have to change them regularly. If IT Security managers had their way then passcodes might be changed on every transaction, whereas users would never want to change them. A working compromise for a single system log-on might be a monthly change, which is still a huge window of opportunity for a hacker. If we remember that there are many systems and merchant sites that a user may need to log into with different credentials then the human-reliant dynamic passcode approach is a non-starter. We, therefore, need some computing technology to help us and if Chip & PIN works for Cardholder present transactions then why not for CNP too? Especially since Issuers have already gone to the expense of distributing secure tokens in the form of EMV payment cards.

The first problem when seeking to exploit our EMV cards in say an Internet access application is that the card obviously has no user-interface, so we need to insert it in something that does. A PC is a prime candidate, however not many of them have in-built smart card readers. As readers today can cost less than eight Euros [15] one solution is to simply issue them to customers, but unfortunately, such attempts often meet with limited success, as customers either have limited IT skills to install the readers correctly, or they simply cannot be bothered to do it. Another solution, which is being given serious consideration by banks, is to issue a small standalone reader device that is relatively easy for the customer to use. Use of the reader plus the EMV card and user PIN, goes under various names including Dynamic Passcode Authentication [16], Chip Authentication Programme (CAP) [17] or simply Token Authentication. In classical security terms it is known as a two-factor [18] security solution, i.e. something you have (the card) and something you know (the four-digit PIN).

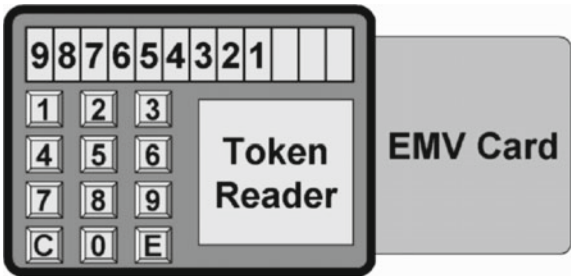
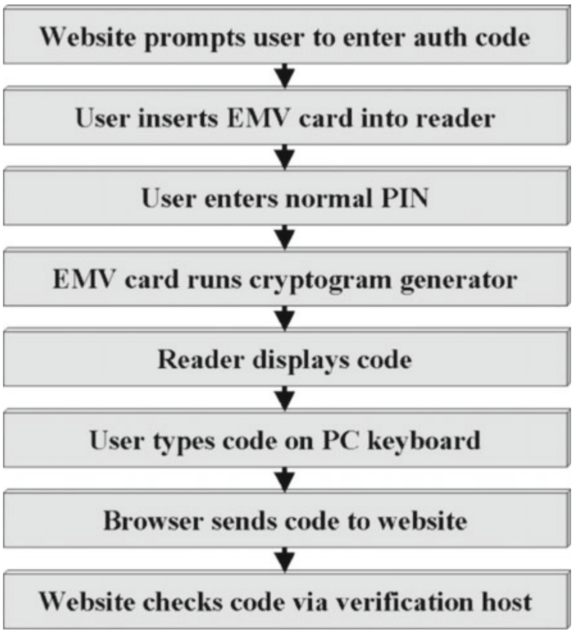


Fig. 5.7 Token reader

Readers can be quite small—not much bigger than the EMV card and an example of a typical device is shown in Fig. 5.7. Using the reader is little more trouble for the user than a normal Chip & PIN transaction and there is no electronic interface to the computer. Information still flows to and from the card, but it is up to the user to visually read and enter the information on the reader keypad and computer keyboard when prompted. The positive aspect of this is that the reader, as an unconnected device, will not be subjected to the normal range of Internet hacking attempts such as spyware, malware etc. As far as the user is concerned the method of use is as shown in Fig. 5.8.

Taking use on the Web as an example, essentially the user inserts their card into the slot, selects the "mode" on a choice of buttons and when prompted on the display enters their PIN. A passcode (typically 8 digits) is then displayed and the



Website prompts user to enter auth code
User inserts EMV card into reader
User enters normal PIN
EMV card runs cryptogram generator
Reader displays code
User types code on PC keyboard
Browser sends code to website
Website checks code via verification host

Fig. 5.8 Using dynamic passcode authentication

user reads this to copy into the appropriate box on the webpage. For an even stronger authentication (proof against pre-play attacks) the website can display a challenge that the user needs to enter as well as their PIN.

The use of devices that generate passcodes for users to type in is not novel and indeed systems such as the Racal Guardata "Watchword" [19] have been around since at least 1992—albeit without a separate card. However, most systems to-date have been proprietary in nature which tends to ultimately limit their roll-out to the population at large. The advantages of leveraging the EMV cards already issued include cost savings, as the cryptographically capable tokens are already in the hands of the users and familiarity of the population with Chip & PIN functionality. The main problem is the cost and distribution of the readers, but since they need no security features and are ubiquitous (any reader will work with any card) the logistics are feasible and many banks have supplied readers to secure e-banking log-on. Customers who have been issued these devices seem to appreciate that their financial institution takes security seriously, but it is probably the case that most readers are kept close to the PC where they are normally used. There are some concerns that borrowing a reader could potentially expose a user to some risk as there may be no proof that it is a legitimate device and whilst this may not result in any violation of the passcode approach, the reader could be used to capture and store sensitive data such as the user PIN. However, this is of limited value as the data read from the chip cannot be used to create a mag-stripe card (the CVV is different) and the card would have to be subsequently stolen, which would be noted by the user and maybe unlikely for a person known well enough to borrow a reader from.

The card itself is not necessarily much different to an EMV card designed without token authentication in mind. There is some added personalisation, data and key management to take care of, but nothing major. This also means that the host systems that normally handle EMV transactions require only minor functional modifications to cope with token authentication. The fact that there is any difference at all is to reduce the number of digits that have to be in the passcode for the Issuer to be able to check the value. To manage this, token authentication cards carry an Issuer Proprietary Bitmap (IPB) that controls how much of the output of the cryptogram makes it into the passcode. At least some bits of the cryptogram (for randomness) and some bits of the ATC (for synchronisation) need to be included. The use of the bit filter is shown in Fig. 5.9. To support cards not specifically personalised for token authentication, all readers are programmed with a default IPB.

The IPB has as many bits as there are in the cryptogram response from the card and each bit acts as a simple switch, e.g. if the IPB bit is set to logic '1' then the cryptogram response bit is retained in the output. As most users are not likely to be too impressed by binary output, there is a binary to decimal mapping prior to display. Typically the passcode is eight or nine decimal digits long.

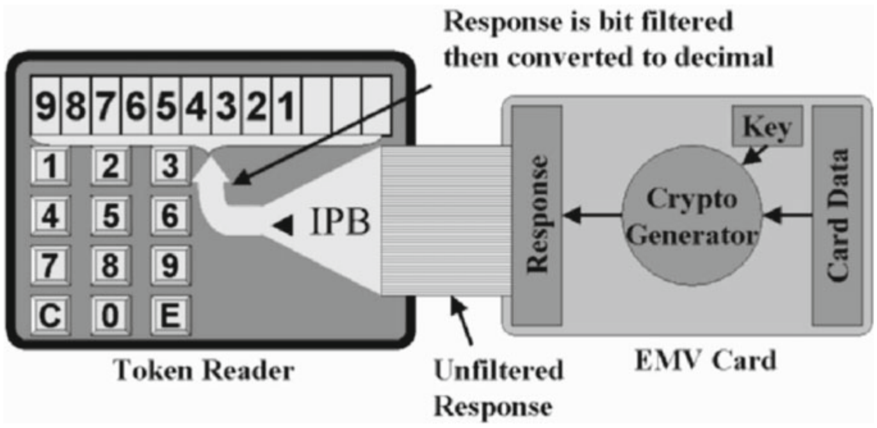


Fig. 5.9 Passcode generation

5.6 Could a Mobile Phone Be a Token Reader?

Technically, it is quite possible for a mobile phone to act as a passcode generator using EMV credentials as many now include the capability to host a virtual card in the Universal Integrated Circuit Card (UICC) or other Secure Element (see Chap. 1) and of course have a keypad for PIN entry. However, there are security concerns relating to whether the operating system handling the keypad is trustworthy. Most instances in mobile devices either use an App which runs a time- based passcode generator (the EMV approach is counter based) or use the mobile as a second channel. In this case, a website requiring authentication will send an SMS to the mobile device with a random code that the user then enters into the appropriate box on the screen—the assumption being that it unlikely that an attacker with stolen ID and password will have real time access to the target's mobile device. The outcome is that there are few instances of mobiles being used as EMV passcode generators.

5.7 Token Authentication Examples

The main use of EMV token authentication is for e-banking log-on. Banks, who have issued cards (normally debit against current accounts) have distributed readers to their customers and require them to be used for access to Internet banking. In some cases it is for any access to the account, but in some others initial access, for say a balance enquiry, is by ID and password and token authentication is only required if money is to be transferred to an external recipient, or new instructions such as standing orders or direct debits are being set up. In addition to passcode generation, with or without a challenge, the specification and readers also allow transaction data, such as amount and destination account number to be entered by the user to be included in the calculations and thus the passcode also acts to prevent manipulation of the sum paid or the destination account.

5.8 Just Wave Your Card to Pay

With the widespread use of contactless travel cards in the transport industry [20] and the development of mobile devices with Near-Field Communication (NFC) technology, the payments industry is also implementing products that allow payment simply by waving a card or mobile device in front of a reader. The products use either DDA or CDA (as in Sect. 5.3.1) to complete an offline transaction with the card needing to be in the reader field for less than 500 ms. Both contactless cards and NFC mobile devices use the ISO/IEC 14443 standard for the communications protocol and although there are some implementation differences, in particular how close the device needs to be to the reader, they can both operate on common reader devices.

For fast transactions, it is of course not realistic for there to be any cardholder authentication such as PIN entry, so implementations have a limit on the transaction amount of typically £30 or so, mostly agreed on a market basis to facilitate Cardholder understanding. Counters control contactless usage and sometimes a Cardholder will be required to complete the transaction using Chip & PIN as a security measure. This will not be apparent to a Cardholder that also uses their card for higher value transactions as Chip & PIN will be occurring from time to time anyway.

Mobile Devices have the potential for the Cardholder to authenticate themselves using a passcode, or in some cases biometrics, before doing the contactless transaction and thus can be used for higher value transactions. There is also the possibility of Issuers managing the card application with over-the-air data communication. Many of these services are new and they may also be linked to mobile wallets,such as Apple Pay [21] and Android Pay [22] which have alternative authentication processes. The whole picture of payments is changing as mobile devices become ubiquitous in the hands of the vast majority of people who currently use payment cards. They are becoming indispensable for many aspects of life and with payment but one small need it should presumably be easy to satisfy in a device with its own data connectivity and much more processing power than a smart card.

5.9 Concluding Remarks

We examined the evolution of Payment Card technologies, from the early concepts, the use of cryptography to protect the initial automation introduced with mag-stripe-based systems and the improvements gained with the smart card replacements as mag-stripe approaches the end of its life. We also examined the opportunities smart card technology allows, including wide scale adoption of PIN for cardholder authentication and the appearance of contactless allowing a fast "Wave & Pay" experience.

We have seen how the technology to secure card present transactions does not transfer directly to CNP transactions and how different technologies are needed to manage e-commerce payments. These include the Internet protocol 3D-Secure—widely deployed, but maybe not as popular with consumers as the industry might wish and two-factor authentication such as token readers that take advantage of the already deployed card base to tightly secure e-banking. Finally we consider that mobile devices, the lifestyle preference for many people, could become the dominant technology for payments.

References

1. Wonglimpiyarat. Strategies of Competition in the Bank Card Business: Innovation Management in a Complex Economic Environment. Sussex Academic Press, 2004.
2. Karl Brinkat, David Main. Smart cards for secure banking & finance. Presentation in the MSc in Information Security, Royal Holloway University of London.
3. Europay-MasterCard-Visa. Emv'96 integrated circuit card specification for payment systems, version 3.0, from https://www.emvco.com/specifications.aspx?id=63, 1996, (accessed: May 24th 2016).
4. EMV Books 1–4 Version 4.1 2004. https://www.emvco.com/specifications.aspx, (accessed: May 24th 2016).
5. Visa, "Secure With Visa-Card Verification Value 2" http://www.visa.ca/en/personal/securewithvisa/cardverify.jsp, (accessed: May 24th 2016).
6. Card Watch "Types of Card Fraud" http://www.cardwatch.org.uk/, (accessed: May 24th 2016).
7. BBC, "Carphone Warehouse in customer data breach", http://www.bbc.co.uk/news/uk-33835185, (accessed: May 24th 2016).
8. Krebs, B., "Hyatt Card Breach Hit 250 Hotels in 50 Nations", http://krebsonsecurity.com/2016/01/hyatt-card-breach-hit-250-hotels-in-50-nations/, (accessed: May 24th 2016).
9. Finextra, "Global Payments breach extends to merchant accounts" https://www.finextra.com/news/fullstory.aspx?newsitemid=23803, (accessed: May 24th 2016).
10. Microsoft, "Antivirus Defense-in-Depth Guide", Microsoft technet 2011, https://technet.microsoft.com/en-us/library/cc162791.aspx, (accessed: May 24th 2016).
11. Mastercard, "Mastercard SecureCode", https://www.mastercard.us/en-us/consumers/features-benefits/securecode.html, (accessed May 24th 2016).
12. Visa, "Verified by Visa", https://www.visaeurope.com/making-payments/verified-by-visa/, (accessed May 24th 2016).
13. Mel H and Baker D., "Cryptography Decrypted" chapter 20 pages 215–227, Addison Wesley ISBN 0-201-61647-5, 2001.
14. Schneier B, "Applied Cryptography", page 170–173, Wiley ISBN 0-471-12845-7, 1996.
15. Xiring "Teo reader", http://www.teobyxiring.com/, (accessed: May 24th 2016).
16. Visa, "Dynamic Passcode Authentication", https://www.visa.gr/media/images/dynamicpasscodeauthentication-42-6506.pdf, (accessed: May 24th 2016).
17. Mastercard, "Mastercard Authentication Solutions", https://www.mastercardconnect.com/mol/molbe/public/login/ebusiness/smart_cards/one_smart_card/biz_opportunity/cap/index.jsp, (accessed: May 24th 2016).
18. Konstantinos Markantonakis, Keith Mayes, Fred Piper, "Smart Card Based Authentication-Any Future", Computers & Security (2005), Elsevier Issue No 24, pages 188–191.

19. Racal Guardata "Watchword datasheet" 1992, http://www.anagram.com/berson/watchword.pdf, https://tfl.gov.uk/fares-and-payments/contactless?intcmp=8257.
20. Transport for London "Contactless" https://tfl.gov.uk/fares-and-payments/contactless?intcmp=8257, (accessed: May 24th 2016).
21. Apple Pay, http://www.apple.com/uk/apple-pay/, (accessed: May 24th 2016).
22. Android Pay, https://www.android.com/intl/en_uk/pay/, (accessed: May 24th 2016).

Chapter 6
Security for Video Broadcasting

Allan Tomlinson and Sheila Cobourne

Abstract This chapter presents an overview of a well-known application of smart card technology, namely that of pay-TV systems. The focus is on the security issues of this particular application and the mechanisms available to meet the security requirements. The chapter begins by establishing the requirements for the application and then looks in detail at the security mechanisms provided by current broadcast standards.

Keywords Smart cards · Pay-TV · Digital Video Broadcast · Conditional Access · Security · Common Scrambling Algorithm

6.1 Introduction

Smart cards have been associated with pay-TV systems for some time now. This chapter will describe this particular application, focusing on the system design constraints and, of course, on the security issues. Although there are many different pay-TV systems in the market, and each one has its own idiosyncrasies and security secrets, the general security problem addressed by these systems is common across all of these applications. It is the general application security that will be discussed rather than the details of any specific solution.

The first objective of this chapter is to provide some understanding of the commercial motivation to provide content protection. A second objective is to show how the overall security solution takes this into consideration to provide a degree of flexibility sufficient to cover as many commercial requirements as possible. The third,

A. Tomlinson
Information Security Group, Royal Holloway,
University of London, Egham, UK
e-mail: Allan.Tomlinson@rhul.ac.uk

S. Cobourne (✉)
Smart Card Centre, Information Security Group, Royal Holloway,
University of London, Egham, UK
e-mail: Sheila.Cobourne.2008@live.rhul.ac.uk

K. Mayes and K. Markantonakis (eds.), *Smart Cards, Tokens, Security and Applications*, DOI 10.1007/978-3-319-50500-8_6

and main, objective is to provide an overview of how a typical digital pay-TV system operates and the mechanisms used to control access to content.

The general security problem alluded to above is that of securely delivering content from a *single source* to thousands, or more often millions, of receivers in such a way that no illegal receivers can access the content. Although data integrity is important, the main security concern is the provision of confidentiality. This problem applies to all broadcast networks—satellite, cable TV, and terrestrial broadcasts—and the mechanisms described in this chapter are applicable to all types of broadcast networks.

All broadcast networks require the establishment and maintenance of a network infrastructure. These networks may require video play-out and scheduling equipment; distribution networks between broadcast sites; and the broadcast network itself, which may require leasing transponders on a satellite. All of this incurs cost to the broadcaster. Another cost is the cost of the content. The broadcaster often has to pay a content provider a licence fee to be able to broadcast the content. In addition to this fee, the licence may place restrictions on the broadcaster requiring the content to be broadcast only at specific times or dates; the number of times it can be broadcast; and perhaps the geographical location of the broadcast. So in addition to the cost of the infrastructure and the cost of the content, there is a cost associated with managing the distribution of the content. The broadcaster's business model relies on recouping these costs, either through advertising, or charging the consumer. Content protection is therefore an important aspect of most digital TV broadcast businesses.

To conclude this introduction it is useful to consider the differences between broadcast networks and the more familiar TCP/IP networks—particularly in terms of the security constraints.

If the networks shown in Fig. 6.1 are compared, the following observations can be made about broadcast networks. First of all, in a broadcast network there is no return path. The communications channel is one-way only, which makes key sharing and key management more difficult than in a conventional network where two-way communication is available. Second, the broadcast network is dealing with

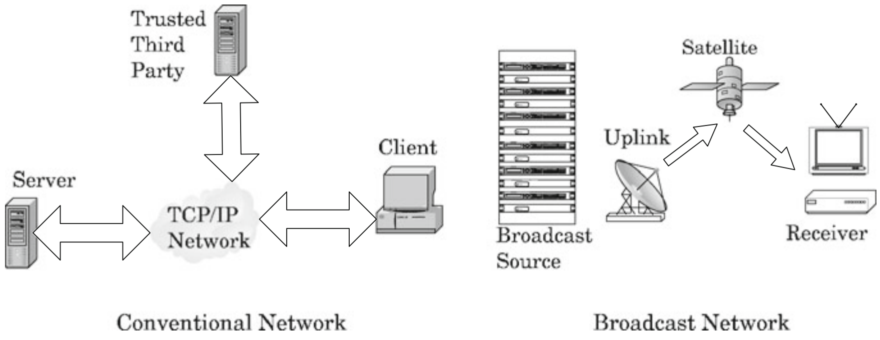


Fig. 6.1 A comparison between broadcast and conventional networks

constantly streaming, often live, video. This creates important constraints on key synchronisation. The keys must be changed at *exactly* the same instant at the play-out centre and at all receivers to avoid temporary loss of service. Finally, there is no way to control who connects to a broadcast network. Thus pirates have access to all broadcast content, without the need to take precautions against being detected by the network operator.

The foregoing has described the environment in which pay-TV systems operate. Before delving into the details of how the security issues are addressed and the role of the smart card, it is first necessary to look in a little detail at the structure of the content we are trying to protect.

6.2 Digital Video Basics

The motivation behind digital transmission is commercial, and lies in the ability to apply digital compression techniques to the signal. Compressing the amount of data that needs to be transmitted provides the ability to broadcast, typically, about five or six digital TV programmes, at VHS quality, in the bandwidth normally used for a single analogue programme. Additional benefits of digital transmission are that more flexible packages of content can be constructed, offering a choice of audio channels for example; the quality of the received signal is often better, up to the point where it fails completely; and finally, it provides the opportunity to secure the content using digital, rather than analogue, techniques.

After converting the analogue video and audio signals MPEG compression [1, 2] is applied to produce digital content. This results in multiple compressed *elementary streams*. In addition to the video elementary stream, there may be one or more audio streams, and perhaps a data stream. This collection of elementary streams is used to reconstruct the programme at the receiver. Several collections of elementary streams representing several programmes are usually combined to form a *digital multiplex* or *transport stream* [3]. When transmitted, this multiplex will occupy the same bandwidth as a single analogue video channel so, as mentioned earlier, it typically carries around five or six programmes in a single channel. The situation is illustrated in Fig. 6.2.

Although the elementary streams are continuously streaming content, in the construction of the transport stream these elementary streams are split into discrete packets. Each packet contains 188 bytes and the stream to which it belongs is identified by a unique Packet ID or PID. Figure 6.2 illustrates how a collection of programmes, or services, may be transmitted. Service 1 represents a standard TV programme with an associated text service; service 2 has multiple audio streams; and service 6 is a simple programme with one audio stream and one video stream.

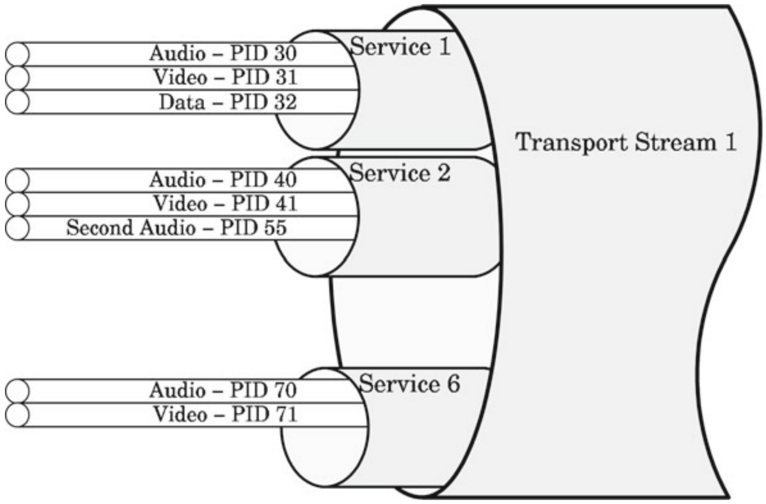


Fig. 6.2 Basic transport stream

6.3 Scrambling

Dividing the elementary stream into uniform packets simplifies the scrambling process. Unlike a continuous stream we now have a well-defined block of data with a fixed length. This means that we can use a block cipher over a fixed block length, or apply a stream cipher to a stream of known length using a key stream re-initialised for each packet.

The scrambling algorithm used is defined in an ETSI standard (ETR 289) [4, 5] and is known as the *Common Scrambling Algorithm* or CSA. At the time of writing[1] there are two versions of this algorithm in use, CSAv1 and CSAv3: this chapter will use the term 'CSA' generically to include both CSAv1 and CSAv3, highlighting any differences in their use where relevant. Table 6.1 summarises the differences between the two versions.

Although both versions of the CSA algorithm have been standardised, the details are subject to a non-disclosure agreement. However, it is widely known that CSAv1 applies both a block cipher and a stream cipher to the payload of the packet. When scrambling, the block cipher is applied first, using cipher block chaining on 8 byte blocks (CSAv1). (CSAv3 is based on two block ciphers—a version of AES128 and "eXtended emulation Resistant Cipher" (XRC) which is a DVB-confidential cipher—and operates on 16 byte blocks.) The result is then scrambled again using a stream cipher based on nonlinear feedback shift registers . The algorithms were designed by a committee of the Digital Video Broadcast[2] (DVB) group. Both the block cipher,

[1]June 2016.

[2]www.dvb.org.

Table 6.1 Comparison of common scrambling algorithm versions

Feature	CSAv1	CSAv3
Key length (CW)	64 bit	128 bit
Encryption algorithm	Proprietary	AES128 variant
		DVB-confidential XRC
Key derivation	Proprietary	subset of IDEA-NXT [11]
		DVB-confidential S-box
Block size	8 bytes	16 bytes
Granularity	1 byte	1 byte
Descrambling gates		approx 100 K
Date	1996	2007

and the stream cipher use the same *Common Key* (CK), hence the name Common Scrambling Algorithm.

Descrambling is the reverse. The stream cipher is applied first, followed by the block cipher. The algorithm was designed to be most efficient when descrambling, since this will be carried out in the consumer equipment where cost is important. A CSAv3 descrambler requires in the order of 100 K gates.

6.4 Synchronisation

For any given elementary stream the Common Key changes rapidly, as often as once every 5 s, and typically around once every 30 s. New keys must therefore be delivered *in advance* of this change, and key changes synchronised *exactly* at transmitter and all receivers. Remember, this is continuously streaming content and any failure to synchronise will result in corruption of the received service.

Another key synchronisation issue arises when channel changes are considered. Typically each continuously streaming programme, or service, is scrambled with a unique key. This leads to a potential problem if someone is "channel surfing"—flicking through each channel, looking for a programme to watch. Even if keys are changed every 5 s no one would be happy to wait this length of time for a new key to be delivered each time they changed channel. So, although the CK may change every 5 s, it must be repeatedly broadcast every 100 ms. or less, to reduce delay when switching channels.

To address the synchronisation problem, it was mentioned that while the current key was being broadcast, the next key is delivered in advance. The DVB standards [6, 7] introduce the the concept of a *crypto-period* as the duration of a valid current key (5 s in our example). The standards also introduce the idea of odd crypto-periods, and even crypto-periods, switching from odd to even (and back) with every key change.

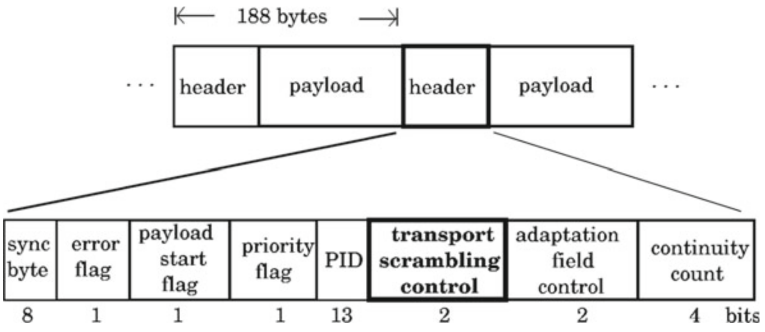


Fig. 6.3 The location of transport scrambling control bits within the packet header

Table 6.2 The Meaning of Transport Scrambling Control Bits.

Bit values	*Description*
0 0	Payload not scrambled
0 1	Reserved
1 **0**	Payload scrambled with **even** key
1 **1**	Payload scrambled with **odd** key

This leads to the notion of having odd and even keys. So while an elementary stream is being scrambled with an odd key, the next even key is being delivered in advance.

Now, for each packet scrambled, an indicator can be set in the packet header to say whether this packet was scrambled with an odd or even key. This indicator is provided by the *transport scrambling control* bits shown in Fig. 6.3. The first bit indicates whether the packet is scrambled or in the clear; the second bit indicates whether the packet was scrambled with an odd or even key. The meaning of the transport scrambling control bits is summarised in Table 6.2.

So now, when a packet arrives at the receiver, if there is a change from odd to even indicated in the packet header, the descrambler will know to use the next (even) key which it should have received in advance of the key change at the transmitter. Thus, key changes can be synchronised precisely without relying on any timers or counters.

The synchronisation process at the receiver may be illustrated with reference to Fig. 6.4. When a packet arrives the receiver inspects the transport scrambling control bits in the header to see if it is scrambled or not. If the packet is scrambled, the receiver then checks whether it has been scrambled with an odd or even key. It then selects the appropriate key, and descrambles the packet.

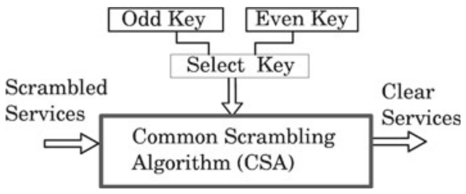


Fig. 6.4 Synchronisation at the receiver

6.5 Key Delivery

The Common Key is cryptographically derived from a *Control Word* (CW) - 64-bit in CSAv1, 128-bit in CSAv3. Since there is no return channel or any means of negotiating the CW, it has to be delivered to each receiver in the broadcast signal itself. Since the CW is broadcast in the transport stream, it has to be encrypted. This encrypted Control Word is delivered in a special stream within the multiplex known as an ECM or *Entitlement Control Message* stream. The structure of an ECM packet is defined by the MPEG standards [3, 7], and it usually carries two encrypted Control Words per message: the current and next. Although the ECM structure is standardised, the algorithms used to encrypt the CW remain proprietary. This allows different vendors to distinguish their products in the marketplace.

Each ECM stream is usually associated with a complete service. In other words, the same Control Word is used to scramble all elementary streams associated with a complete service. The standards do however allow an ECM to be associated with single elementary streams too. This allows, for example, an additional audio stream, perhaps a second language, to be encrypted separately from the main service. Thus

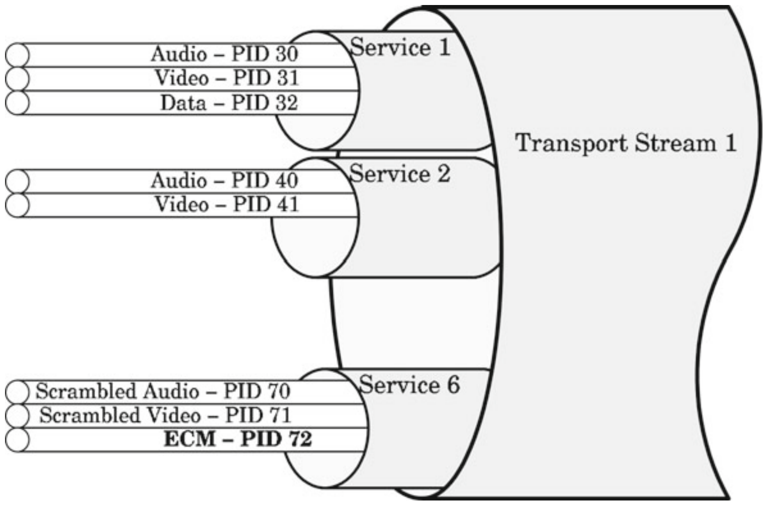


Fig. 6.5 ECM stream

the broadcaster is able to provide this second audio service only to subscribers who have paid for it. ECMs may also be associated with groups of services if cost is an issue for the broadcaster. Grouping services in this way could reduce the need for expensive specialist encryption hardware. However, most broadcasters use one ECM stream per service as illustrated in Fig. 6.5 where service 6 has now been scrambled.

6.6 Access Requirements

ECM messages also contain *access requirements*. These access requirements, also known as access criteria, identify what "packages" the user must be subscribed to before they can access the service. For example, a football match might require that the user has paid their subscriptions to the "sports package" before they can view the programme. Access requirements may also place a constraint on the geographical region that a receiver is in. Again to use the football example, there may be a restriction on viewing a football match in the city where the game is being played. Each receiver therefore has a set of viewing rights, depending on the subscriptions paid. Only those receivers with rights that match the access criteria are able to recover the CW from the ECM and view the programme.

Before any attempt is made to decipher the CW delivered in the ECM, the access criteria are checked and compared with the *rights* that the receiver has. These rights are held in secure storage on the receiver, typically on a smart card, and may be updated by the service provider depending on the subscriptions paid by the user. Thus each receiver may have a different set of rights, giving access to different sets of services.

The rights that each receiver has are updated by the service provider by sending a second type of entitlement message: the *Entitlement Management Message*, or EMM. This type of entitlement message is different from the ECM because where the ECM is associated with a service and must be delivered to all receivers, the EMM is associated with an individual receiver and the rights delivered should not be available to all receivers in the broadcast network. The way this is managed is by means of a key hierarchy as discussed in Sect. 6.7. As was the case with the ECM, the EMM structure is standardised [3, 7], but the algorithms used to encrypt the contents remain proprietary.

The majority of pay-TV services are managed on a subscription basis. That is, the user subscribes to a set of services for which a monthly fee is paid for continuous access to a set of broadcast services. A more flexible mechanism to manage payment for services is to allow subscription on a per-programme basis. This is known as pay-per-view (PPV). In "Call Ahead PPV" the user calls the service provider in advance of the programme being broadcast and buys a subscription for that programme only, for example a sporting event such as a football match. Once payment has been made, the broadcaster will send an EMM to the individual receiver to update its rights. A more convenient mechanism is "Impulse PPV" where instead of calling the service provider each time, the receiver contains a certain amount of credit which is used to

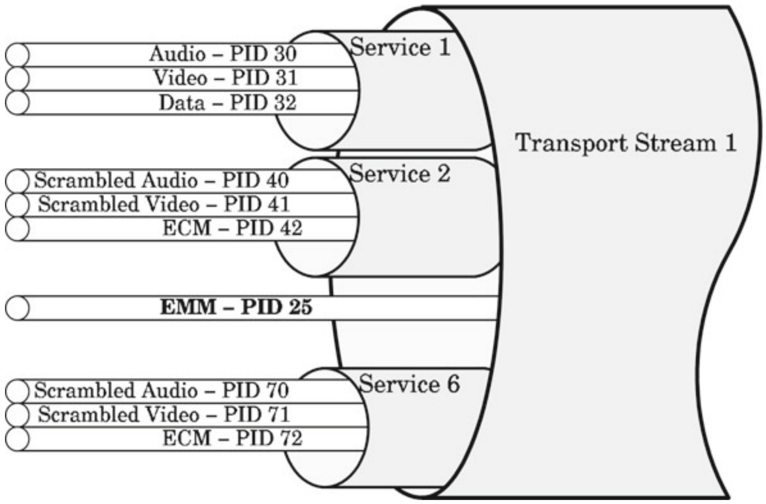


Fig. 6.6 EMM stream

pay for such PPV services. Thus if the receiver is in credit, the event can be viewed, and the credit level adjusted accordingly.

The model of service delivery may now be updated to include the EMM stream as shown in Fig. 6.6.

6.7 Key Hierarchy

In Sect. 6.5, we saw that the ECM message delivered an encrypted CW. If the CW is encrypted under the control of a key K_0, $ECM = CW_{K_0}$, then it is clear that all receivers that are authorised to view the programme must have K_0. K_0 however is not permanent, it may be changed on a per-programme basis, or per day, or week, depending on the service provider's requirements. When K_0 is changed, the new key must be delivered to all receivers. This is accomplished by another type of EMM message which, instead of delivering rights, delivers keys to the receivers.

Of course the keys being delivered by the EMM must be encrypted before being broadcast. Thus K_0 will be encrypted under a more long-term key, say K_1. Each receiver could then have its own unique K_1 and a unique EMM generated and broadcast for each receiver. In large networks, this may be impractical since the bandwidth required to continuously broadcast individual EMM messages for a period of time long enough to ensure all receivers are updated, will become too large, especially if K_0 is changing at a relatively high rate, say once per programme.

In practice receivers are often organised into groups so that they can be updated on a group basis rather than an individual basis. Each group will share K_1, which

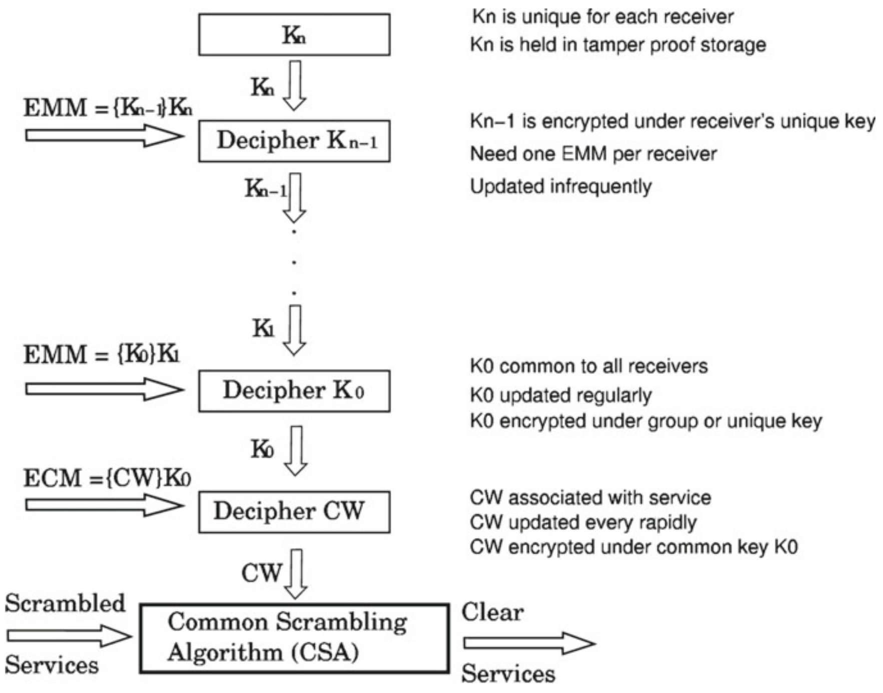

Kn
Kn is unique for each receiver
Kn is held in tamper proof storage
Kn
EMM ={Kn-1}Kn
Decipher Kn-1
Kn−1 is encrypted under receiver's unique key
Need one EMM per receiver
Updated infrequently
Kn-1
K1
EMM = {K0}K1
Decipher K0
K0 common to all receivers
K0 updated regularly
K0 encrypted under group or unique key
K0
ECM ={CW}K0
Decipher CW
CW associated with service
CW updated every rapidly
CW encrypted under common key K0
CW
Scrambled
Services
Common Scrambling Algorithm (CSA)
Clear
Services

Fig. 6.7 Key hierarchy

will be delivered to each group member encrypted under another key, say K_2 which this time may be unique to each receiver.

Thus we have a key hierarchy as illustrated in Fig. 6.7. In a broadcast network it is not practical to duplicate scrambled content, scrambling it with keys unique to each receiver. Therefore at the bottom level, the content itself must only be scrambled once under one key: the CK derived from the CW. This CW changes rapidly and is delivered in an ECM associated with the scrambled service. Higher level keys change less rapidly and are delivered in EMM messages. The longer update interval allows more time to ensure all receivers are updated with new keys. Ultimately, at the top of the key hierarchy, all receivers are updated one at a time with keys encrypted under the *unique key* for that receiver.

6.8 Implementation

Prior to standardisation, all aspects of the previously described architecture were proprietary. The DVB standards body however defined specifications [4–7] which, if followed, will allow scramblers and receivers to operate with *any* proprietary system. These standards isolate scrambling functionality from higher layers of the

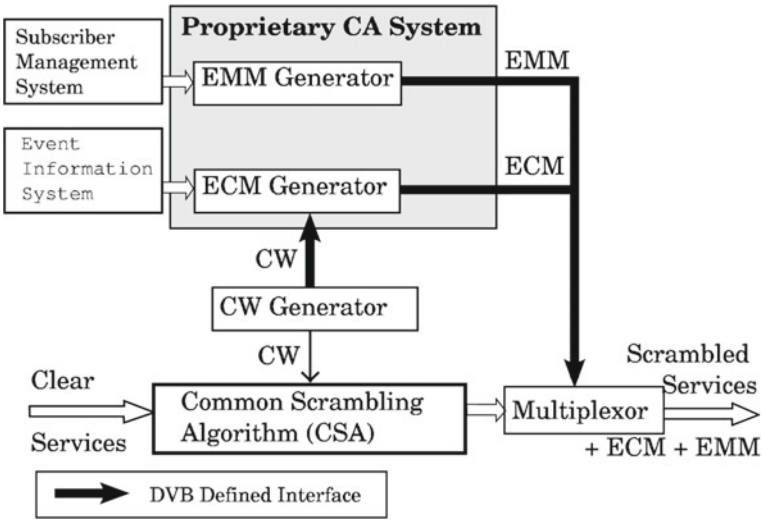


Fig. 6.8 Scrambling at the broadcast centre

key hierarchy, and define the scrambling algorithm to be used. The standards also define the interface between the scrambler and higher layers of the key hierarchy. Thus the higher layers can still remain proprietary allowing vendors to design their own systems, known as *Conditional Access* (CA) systems, while at the same time allowing broadcasters to switch CA systems if they require to do so.

In addition to the commercial benefit of defining the interface between proprietary and standard parts of the key hierarchy, there is also a security gain. If the whole key hierarchy was standardised then there would be greater incentive for pirates to attack the system. Whereas with the standard–proprietary split, a successful attack may compromise one CA system, but the others will remain secure. Moreover, the ability to choose from a number of CA systems provides an incentive to the CA vendors to constantly monitor and improve their systems if they are to compete in an open marketplace.

Figure 6.8 illustrates how content is scrambled at the broadcast play-out centre [7]. The CW used to scramble the content is passed to the CA system and used to construct the ECM messages. An Event Information System is often used to schedule when to apply scrambling and what access criteria to use. The CA system also generates the EMM messages and delivers them to individual receivers or groups of receivers based on data held in a Subscriber Management System. The ECMs, EMMs, and scrambled content are then combined to form the digital multiplex prior to transmission on the broadcast network.

The descrambling process at the receiver is illustrated in Fig. 6.9. The *Common Interface Module,* or CIM is a PC Card module that the user may plug into a compatible receiver. The interface between this module and the receiver is defined by a CENELEC standard [6], and the CIM also has a standard ISO smart card interface.

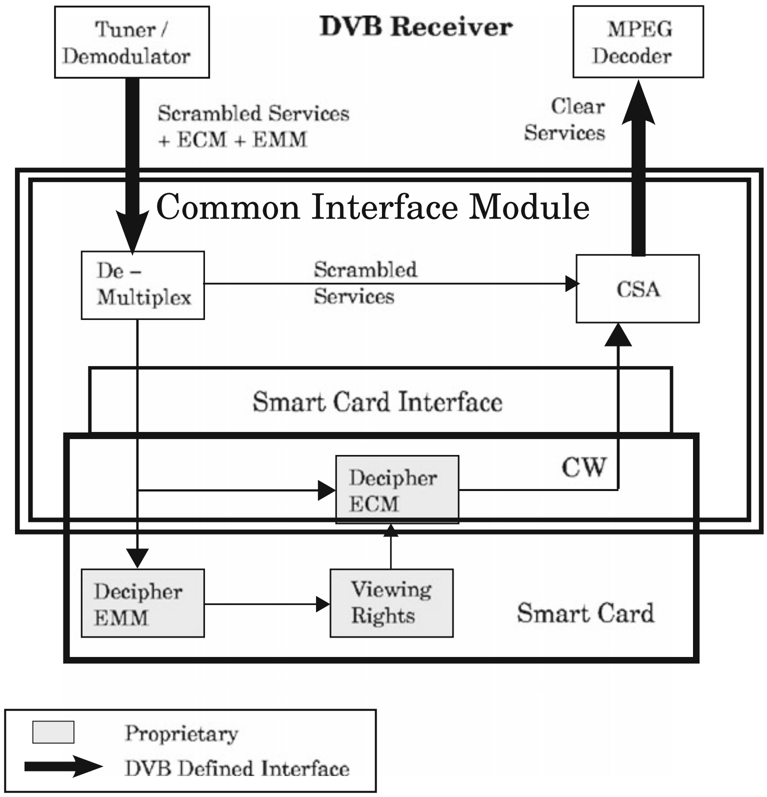


Fig. 6.9 Descrambling at the receiver

The CIM typically contains the hardware to descramble the content in real time, and a smart card, initialised with the unique key at the top of the key hierarchy, is typically used to recover and store higher level keys. Thus the smart card stores the high level keys in a tamper resistant location and generally only provides the CW to the CIM as required. This hardware architecture raises a potential security issue: since the interface between the smart card and the CIM is standardised, it is vulnerable to eavesdropping. In other words, an attacker who is familiar with the CENELEC standard will have access to the Control Words as they pass across this interface. To counter this vulnerability a shared key is often established between a smart card and the descrambling hardware, and used to set up a secure communications channel to protect the CW in transit.

Therefore the CIM and smart card will be unique to each different type of Conditional Access system, implementing not only the content descrambling algorithm, the CSA, but the algorithms used throughout the key hierarchy. The exact details of these algorithms remain proprietary to each Conditional Access system vendor. The smart card, in addition to securely storing the high level keys, will also implement

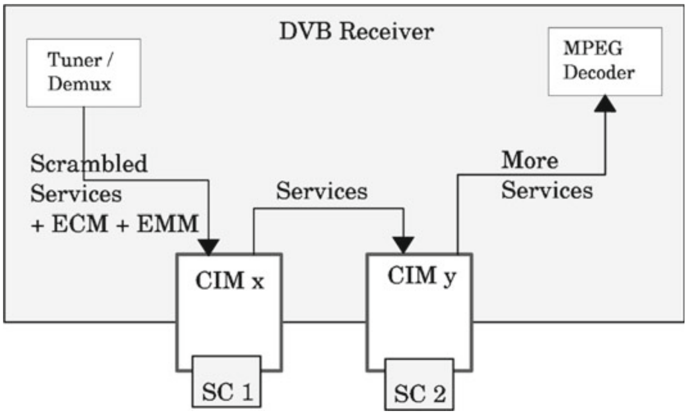


Fig. 6.10 Daisy chaining CIMs

some of these proprietary algorithms. The division of processing between smart card and CIM will depend on the particular CA system in use.

The receiver, once tuned to a particular frequency, will receive the complete digital multiplex containing all scrambled services, ECM and EMM messages. This data stream will be de-multiplexed, based on packet ID to reconstruct the elementary streams. The EMM stream will be processed to recover viewing rights and keys delivered to that particular receiver. These rights and keys will typically be stored on a smart card. When a viewer selects a service to watch, the corresponding ECM stream will be de-multiplexed, processed by the smart card and any Control Words returned to the descrambler. Thus the CIM will be able to descramble the content in real time.

The advantage of providing a standard interface at the receiver is that a CIM for one Conditional Access vendor may be replaced by a CIM for another vendor. This gives the consumer wider access to services without the need for multiple receivers. Some receivers have multiple CIM slots, allowing modules to be daisy chained as illustrated in Fig. 6.10. This provides greater user convenience, since the modules will not need to be swapped to receive services scrambled by multiple service providers with different CA systems.

Taking user convenience one step further, and removing the need for a hardware CIM altogether led to the idea of emulating the CIM in software. This is the concept behind the DreamBox from Dream Property GmbH[3] [8], which is a sophisticated DVB receiver that has much of the functionality of a powerful PC, including an Ethernet interface. This type of receiver supports the implementation of the CIM in software, with the aim of providing as much flexibility as possible to the user. However, this opens up the possibility of cloning a genuine CIM in software and distributing this over the Internet to many illegal receivers. Since the CIM is not a

[3]http://www.dream-multimedia-tv.de.

tamper resistant device it may be possible to extract any secret key that is shared with a genuine smart card. Moreover, a genuine receiver may be modified to recover the Control Words from its CIM and distribute these over the Internet too. The pirate receivers will then be able to recover the CW and illegally access the scrambled content. If a smart card is completely reverse engineered, it too could be simulated on the DreamBox. Details of these attacks together with proposed safeguards based on behavioural analysis built into a smart card are described by Francis et al. [9]. An alternative may be to use trusted computing technology to ensure that the host platform has not been tampered with before the descrambler is enabled [10].

At the transmission site, the DVB standards define the interface between the scrambler and the Conditional Access system [7]. This allows broadcasters to change CA systems if they desire, or to use multiple CA systems simultaneously. The latter is referred to as "simulcrypt" and is illustrated in Fig. 6.11.

In Fig. 6.11, although the service is only scrambled once, with one CW, the standard interface allows this CW to be passed to *two* Conditional Access systems, CAS 1 and CAS 2. Thus *two* ECM message streams are generated for each scrambled service, both containing the encrypted CW, but the CW will be encrypted with a different algorithm and a different key in each ECM. In addition to the two ECM streams, there will be a second EMM stream to manage entitlements controlled by the second CAS as shown in Fig. 6.12. The benefit this brings to the broadcaster is that they can now provide services that can be accessed by receivers designed with two different Conditional Access systems, thus giving access to a wider potential market.

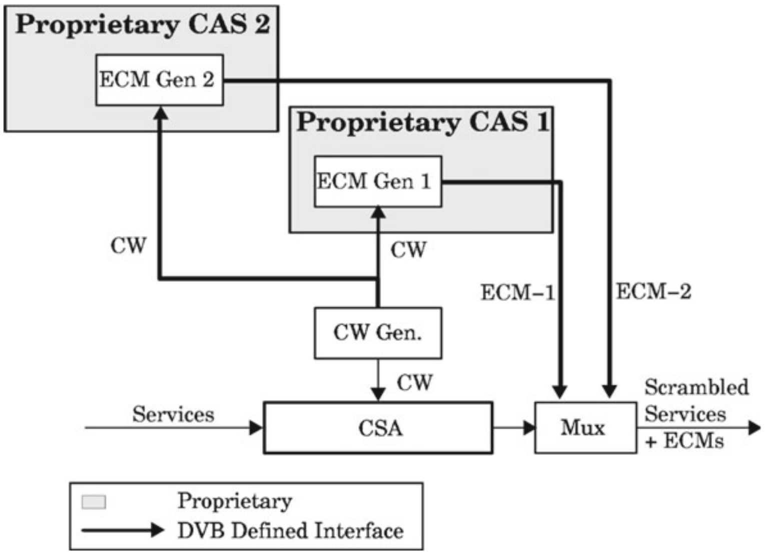


Fig. 6.11 Simulcrypt

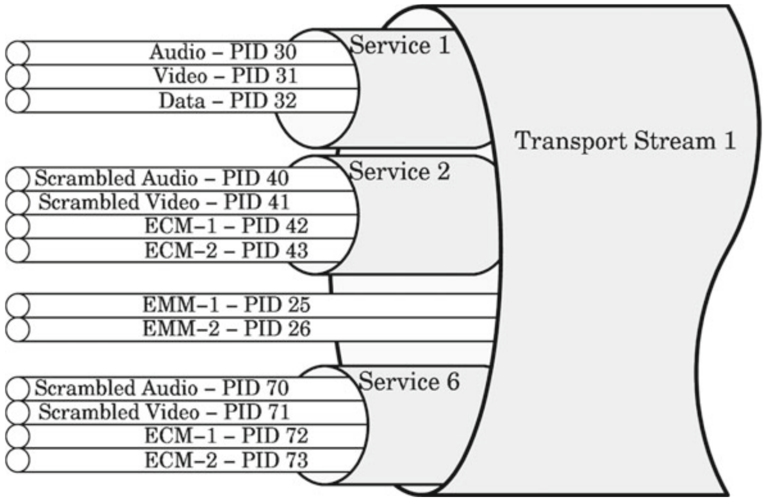


Fig. 6.12 Simulcrypt transport stream

6.9 In Conclusion

In this chapter, the construction of a digital TV multiplex has been briefly described. As far as security is concerned, the services carried in this multiplex may be scrambled by a standard "Common Scrambling Algorithm", CSA [4, 5], using a key derived from a Control Word, CW. This Control Word is delivered to receivers in an ECM message which also contains access criteria. Changes in CW are synchronised by setting bits in the packet header to indicate which CW was used to scramble the packet. Since the CW must be encrypted before being broadcast another key must be made available to the receiver to allow recovery of the CW from the ECM. This key is delivered in an EMM message. Again, this key must be encrypted, which leads to a key hierarchy. At the top of the key hierarchy is a key *unique* to each receiver which will be stored in a tamper proof area on a smart card. The EMM message is also used to deliver rights to the subscriber, which again must be stored in tamper proof memory. These rights are compared with the access criteria in the ECM before any attempt is made to recover the CW.

Although the bulk encryption, or scrambling of content, uses a standard algorithm, protection of the higher level keys remains proprietary. This led to two standards being developed by DVB to allow interoperability. At the receiver the "Common Interface" standard [6] defines the interface between a PC Card Common Interface Module and the receiver. This allows the consumer to swap Conditional Access systems by swapping CIM modules. At the transmission site, the "Simulcrypt" standard [7] defines the interface between the scrambler and the Conditional Access system. Not only does this allow the broadcaster to swap systems, if required, but it allows the broadcaster to use several systems simultaneously, thus providing access to more receivers.

In any secure system, some residual vulnerabilities remain, and the Pay-TV system described here is no exception. Many attacks on Pay-TV systems exploit vulnerabilities in smart cards to create clones. Moreover, with the advent of the DreamBox, the CIM can also be cloned and Control Words distributed over the Internet. The size of the key used with CSAv1 is only 64 bits: although this version of CSA is not recommended for new deployments, legacy installations may still be using this short key length and not benefitting from the security advantages of the later CSAv3. However, this vulnerability may be mitigated to some extent by frequent changes of the CW. A final vulnerability is the fact that proprietary mechanisms are used to manage the higher level keys in the Conditional Access systems. Since these mechanisms have not been subject to public scrutiny little can be said about their security except that some are, no doubt, more secure than others

References

1. ISO/IEC, "Information Technology – Generic Coding of Moving Pictures and Associated Audio: Audio," International Standard ISO/IEC 13818-3, International Organization for Standardization (ISO), Geneva, Switzerland, 1994.
2. ISO/IEC, "Information Technology – Generic Coding of Moving Pictures and Associated Audio: Video," International Standard ISO/IEC 13818-2, International Organization for Standardization (ISO), Geneva, Switzerland, 1995.
3. ISO/IEC, "Information Technology – Generic Coding of Moving Pictures and Associated Audio: Systems," International Standard ISO/IEC 13818-1, International Organization for Standardization (ISO), Geneva, Switzerland, 1995.
4. ETSI, "Digital Video Broadcasting (DVB); Support for use of Scrambling and Conditional Access (CA) within Digital Broadcasting Systems," ETSI Technical Report ETR 289, European Telecommunications Standards Institute (ETSI), Sophia Antipolis, France, Oct. 1996: available at http://www.etsi.org/deliver/etsi_etr/200_299/289/01_60/etr_289e01p.pdf accessed June 2016
5. ETSI, "Digital Video Broadcasting (DVB); Support for use of the DVB Scrambling Algorithm version 3 within Digital Broadcasting Systems," ETSI Technical Specification ETSI TS 100 289 V1.1.1 (2011-09), European Telecommunications Standards Institute (ETSI), Sophia Antipolis, France, 2011: available at http://www.etsi.org/deliver/etsi_ts/100200_100299/100289/01.01.01_60/ts_100289v010101p.pdf accessed June 2016
6. CENELEC, "Common Interface Specification for Conditional Access and other Digital Video Broadcasting Decoder Applications," CENELEC Standard 50221, European Committee for Electrotechnical Standardization (CENELEC), Brussels, Belgium, Feb. 1997.
7. ETSI, "Digital Video Broadcasting (DVB); Head-End Implementation of DVB Simulcrypt," ETSI Standard TS 103 197 V1.5.1 (2008 - 10), European Telecommunications Standards Institute (ETSI), Sophia Antipolis, France, 2008: available online at http://www.etsi.org/deliver/etsi_ts/103100_103199/103197/01.05.01_60/ts_103197v010501p.pdf accessed June 2016
8. D. Multimedia TV GmbH, Dreambox DM7000S user manual, 2004: available online at http://www.dream-multimedia-tv.de/en/downloads accessed June 2016
9. L. Francis, W. Sirett, K. Markantonakis, and K. Mayes, "Countermeasures for attacks on satellite TV cards using open receivers," in Australasian Information Security Workshop 2005 (AISW2005) (R. Buyya, P. Coddington, P. Montague, R. Naini, N. Shepperd, and A. Wendelborn, eds.), vol. 44, pp. 53–158, Australian Computer Society Inc., 2005.

10. E. Gallery and A. Tomlinson, "Conditional Access in Mobile Systems: Securing the Application," in The First International Conference on Distributed Frameworks for Multimedia Applications DFMA 05, (Los Alamitos, CA, USA), pp. 190–197, IEEE Computer Society, February 2005.
11. MediaCrypt AG, "IDEA NXT Technical Description," available online at https://web.archive.org/web/20070928014200, http://www.mediacrypt.com/_pdf/NXT_Technical_Description_0406.pdf

Chapter 7
Introduction to the TPM

Allan Tomlinson

Abstract The Trusted Platform Module (TPM) and smart card devices have many features in common. Both are low cost, tamper resistant, small footprint devices used to provide the basis of a secure computing environment. This chapter presents an introduction to the security mechanisms provided by the TPM highlighting those not typically found on a smart card. The concept of "ownership" is one of the major differences between the TPM and a smart card and this is described in some detail before concluding with a review of some of the security services uniquely provided by the TPM and a description of some recent changes to the TPM standard.

Keywords Smart cards · Trusted computing · TPM · Security

7.1 Introduction

Smart cards provide a wide range of functionality from simple storage media to complex processors. A common design goal across this diversity however, is the provision of some degree of secure processing, implemented in secure hardware. The Trusted Platform Module, or TPM, is similar to a smart card device in that it is a small footprint low cost security module typically implemented as a tamper resistant Integrated Circuit (IC). The TPM however, has been specifically designed to be a building block for trusted computing. So although there are many similarities, the different design goals have resulted in a number of differences between the two types of device. One major difference is that the TPM is considered to be a fixed token bound to a specific platform, whereas a smart card is a portable token traditionally associated with a specific user across multiple systems. This is not to say that the two technologies are mutually exclusive, but rather that they may be considered as complementary.

The details of smart card architectures and applications are discussed elsewhere in this book so this chapter will focus on the TPM and expand on the complementary aspects of this device, describing the specific mechanisms where the TPM differs

A. Tomlinson (✉)
Information Security Group, Royal Holloway,
University of London, Egham, UK
e-mail: Allan.Tomlinson@rhul.ac.uk

K. Mayes and K. Markantonakis (eds.), *Smart Cards, Tokens, Security and Applications*, DOI 10.1007/978-3-319-50500-8_7

from a typical smart card IC. In order to fully understand the rationale behind these mechanisms, it is important to understand the underlying design goals of trusted computing that guided the development of the TPM.

This chapter therefore begins with some background on trusted computing to explain the features that the TPM is designed to support and the rationale behind the design decisions that were made. The fundamental features of the TPM that emerged from these design constraints are described in Sect. 7.3. This section focuses on the functionality of the basic TPM building blocks to expose some of the differences between the design of smart card ICs and the TPM. Many of the differences that appear, arise from the contrasting requirements for ownership and management between the two types of device. Section 7.3 therefore looks at these concepts in some detail. The differences between the two types of device result in differences in functionality, and Sect. 7.4 looks at some of the security services not found in typical smart card devices, which are fundamental to the operation of the TPM. We conclude by noting some of the recent updates to the TPM specification.

7.2 Trusted Platforms

Where smart cards may be considered as general purpose security processors, the TPM has been designed specifically to support trusted computing platforms. Therefore, in order to understand the TPM design requirements, it is first necessary to understand what the desirable features of a trusted platform are. To do this, a definition is required as to exactly what is meant by the term “trusted platform”.

The concepts of trusted computing, and a Trusted Computing Base, or TCB, are not new and are described in many publications ranging from the Orange Book [1] through to more recent material that describe these ideas within the context of contemporary developments in computer security [2–4]. One such development is the emergence of the Trusted Computing Group[1] (TCG) that has attempted to define what is meant by a trusted computing platform and that has produced a series of standards that can be used to design trusted platforms. The TCG defines trust to be “the expectation that a device will behave in a particular manner for a specific purpose” [5]. The TCG’s definition of a trusted platform therefore, is a platform that behaves in such a manner. This is, necessarily, a rather high-level definition: but what this means is that any entity that interacts with such a trusted platform can be given some degree of assurance that the platform (system or computer) will behave in the way that the entity expects it to. Providing assurance of expected behaviour does not in itself provide any security. To achieve that, the entity relying on this assurance still has to ascertain that the “expected behaviour” is indeed secure. However, assuming that the relying entity is satisfied that the expected behaviour is secure, then he may, for example, be assured that any data given to the system is kept confidential, or that no malware is running on the platform.

[1]https://www.trustedcomputinggroup.org.

There are many ways to design trusted computing platforms that allow statements about expected behaviour to be made [4], but the approach taken by the TCG is to have some physically secure trusted component that can be used as a foundation upon which trust in the rest of the system can be built. The purpose of the TPM is to provide this foundation. For this chapter, a more appropriate definition of a trusted platform is provided by Pearson [6] who states that

> A Trusted Platform is a computing platform that has a trusted component, probably in the form of built-in hardware, which it uses to create a foundation of trust for software processes

or Balacheff et al. [7] who say

> A trusted platform (TP) is defined as a computing platform that has a trusted component, which is used to create a foundation of trust for software processes.

It is perhaps appropriate at this point to make a subtle distinction between what is meant by a *trusted component,* such as the TPM, and a *trustworthy component.* One generally accepted definition of these terms is given by Anderson [8, 9] who states that

> The proper definition is that a *trusted* system or component is one whose failure can break the security policy, while a *trustworthy* system or component is one that won't fail

By implementing this trusted component, the TPM, as a tamper proof IC; and binding it to the platform, usually on a printed circuit board containing a more powerful processor capable of running software applications; the TPM can be used as the foundation of trust for higher level processes that run on the main processor.

7.2.1 *Fundamental Features of a Trusted Platform*

In order to establish this foundation of trust, the TPM is expected to provide a fundamental set of security features which have been defined by the TCG. The minimum set of features that a trusted platform should have are: protected capabilities; integrity measurement; and integrity reporting [5]. Providing support for these features leads to the definition of the security requirements of the TPM.

7.2.1.1 Protected Capabilities

To meet the requirements of a trusted platform, according to the TCG [5], the system should provide some form of protected capabilities. In the TPM design principles specification, the concept of protected capabilities is used to "distinguish platform capabilities that must be trustworthy" [10].

These trustworthy protected capabilities are abilities to execute a command or set of commands on the TPM which access *shielded locations* where it is safe to operate on sensitive data [5]. Examples of protected capabilities in the TPM include protection and reporting of integrity measurements (described below); and storage and management of cryptographic keys.

7.2.1.2 Integrity Measurement and Storage

Any trusted platform will need some means of measuring how it is configured and what processes are running on that platform. The increasing complexity of personal computers and software applications has resulted in an increase in the number of processes that run on a typical PC. Many of these processes are launched implicitly, rather than explicitly by the user. Under such circumstances it is difficult to tell if the code being executed on a particular platform is a legitimate process or not, and consequently, if this particular platform can be trusted. It is important therefore, if a platform is to be trusted, that it has some means of measuring the integrity of the processes it is running. This measurement should result in some form of *integrity metric* [5] which Pearson defines as "a condensed value of integrity measurements" [11]. This integrity metric can then be compared with acceptable values for a trusted platform. Having obtained an integrity metric for the platform it is often useful to store that value somewhere for later use. Of course such data needs to be held in secure storage locations such as would be provided by the protected capabilities described above.

In measuring platform integrity in this manner there has to be a starting point. It may be acceptable for a particular application to perform integrity measurements provided that the application itself is trustworthy. Even if the operating system can verify the integrity of this application, the integrity of the operating system too has to be verified. Ultimately there will be some integrity measurement process that exists which cannot itself be verified. This process is known as the Root of Trust for Measurement, or RTM. This is the starting point in the chain of integrity measurements, and ideally this process should run on tamper proof hardware, with the execution code being stored in secure storage.

7.2.1.3 Integrity Reporting

The third requirement for a trusted platform [5] is that it should be able to report its configuration to a challenger who requires this information in order to decide how much trust to place in the platform. In other words, the platform should have some means to report its integrity metrics and to vouch for the accuracy of this information. This attestation requires the generation of evidence that the challenger can rely on in making its trust decision. The implication here is that the integrity metrics can be signed by the trusted platform and that the challenger has a certificate that can be used to verify the signature.

7.2.2 Additional Features

While the above features would allow the implementation of a basic trusted platform as defined by the TCG, there are other features that would be required to create a more flexible implementation of a trusted platform. These include: confidentiality and integrity protection; secure storage; and process isolation.

7.2.2.1 Confidentiality and Integrity Protection

In addition to the protection of integrity metrics, a trusted platform should provide both confidentiality and integrity protection to any data as required by the user. This could include both user data, and application code. Moreover, these security services should be available to protect this data while it is being stored, and during the execution of any process.

7.2.2.2 Secure Storage

The provision of confidentiality requires that a trusted platform is able to encrypt any user data for secure storage. Access to the data is controlled by a securely stored cryptographic key. Similar to encryption is the concept of sealing. In this case access to the data is controlled by platform state, the use of a cryptographic key is optional. This means that data can be sealed to a set of integrity metrics that reflect the platform state. The data will then only be accessible when the platform is in the desired configuration. The sealing mechanism can be used, for example, to ensure that no rogue applications are running before access is granted to sensitive data.

7.2.2.3 Process Isolation

The secure storage and integrity checking mechanisms will protect data during storage. To protect data during execution the provision of process isolation is necessary. The concept here is to provide an isolation kernel between the hardware and any operating system. This isolation kernel is used to create and manage multiple secure compartments. These compartments exist in parallel, on the same machine, and access is managed exclusively by the isolation kernel. Each compartment can then run its own operating system and applications in isolation from any other processes that are executing in parallel. In this way, application code and data can be protected during execution. Furthermore, the use of an isolation kernel greatly simplifies the validation of acceptable integrity metrics: by isolating processes, the set of acceptable platform configurations can be reduced to one operating system and one application only.

7.3 TPM Features

The previous section described the main security requirements for the TPM and the rationale behind these requirements. These high-level requirements for a trusted platform provided much of the input to the development of a series of standards which are published by the TCG, including the specification of the TPM. This specification has gone through two major changes: from v1.1b to v1.2 in 2003, and from v1.2 to v2.0 in 2014.

In order to gain an understanding of the TPM functionality, we begin by discussing the features of TPM 1.2. We will consider the enhancements made to version 2.0 in Sect. 7.5. Both versions of the specification are written as a collection of documents. For version 1.2 the documents describe the TPM design principals [10], which will be reviewed in more detail in the this section; and the data structures [12] and commands [13] used to control the TPM. A discussion of TPM structures and commands is beyond the scope of this chapter but may be found in other work [14, 15].

7.3.1 TPM Components

When comparing the TPM to smart cards, an important point to note is that the TCG specifications do not mandate that the TPM is implemented as an IC. The standards define TPM *functionality,* leaving it open for developers to implement this functionality as they wish, either in hardware or software. Having said that, most commercially available implementations of the TPM are hardware-based and produced by the major IC manufacturers. The main building blocks of the TPM are shown in Fig. 7.1.

The TPM itself, however implemented, must be physically protected from tampering. In PCs, for example, this may be accomplished by physically binding the TPM to the motherboard to make it difficult to disassemble or transfer to a different platform. It is not a requirement for the TPM to be tamper proof, but tamper resistant. The TCG also require that tamper evidence measures should be deployed to enable detection of any tampering by physical inspection.

It is important to note when looking at Fig. 7.1, that the TPM standards do not specify the communications interfaces or bus architecture, leaving these decisions to be made by the developers.

7.3.2 I/O Block

The TCG do, however, specify an interface serialisation transformation that allows data to be transported over virtually any bus or interconnect. This is one of the functions of the I/O block. This block manages information flow between the components

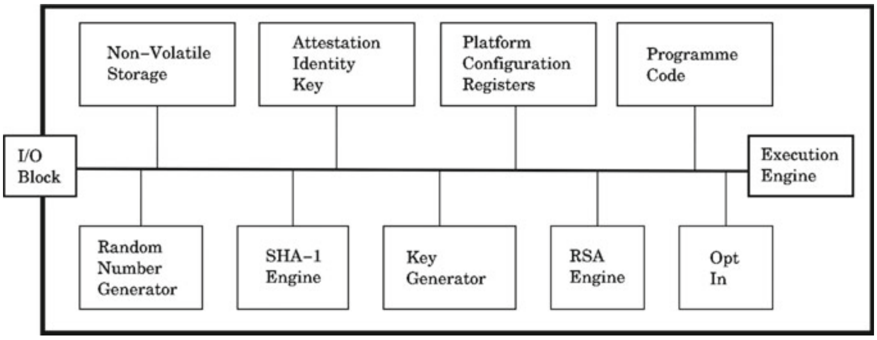


Fig. 7.1 TPM building blocks

illustrated in Fig. 7.1, and between the TPM and the external bus. Since the I/O block is managing data flow, it is also able to control access to the various TPM components. The access rights are determined by flags maintained by the Opt-In block.

7.3.3 Non-Volatile Storage

As with a smart card IC, a TPM has some non-volatile memory to store long-term keys. Two long-term keys are stored in non-volatile memory on the TPM. The first of these is the Endorsement Key (EK); the second key is the Storage Root Key (SRK) which forms the basis of a key hierarchy that manages secure storage.

The TPM also uses non-volatile memory to store *owner* authorisation data. This authorisation data is, in effect, the owner's password and is set, not by the manufacturer, but during the process of taking ownership of the TPM. There are also a number of persistent flags related to access control and the Opt-In mechanism, discussed in Sect. 7.3.12, that need to be stored in non-volatile memory.

7.3.3.1 Endorsement Keys

The Endorsement Key (EK) is a fundamental component of a TPM. It is also something unique to the TPM which is not found on a smart card device. It is therefore important to consider how the TPM uses this key in a little more detail.

For the TPM to operate, it must have an endorsement key embedded in it. To be more precise, it must have an endorsement key pair, of which the private key is embedded in the TPM and *never leaves* it. The public EK is contained in a certificate and is only used in a limited number of procedures. The reason for limiting the use of the EK certificate is because the EK is unique to each TPM and consequently may be used to identify the device, and by extension the platform. Therefore, to protect user privacy when interacting with other entities, the use of the EK is restricted and internally generated aliases, the Attestation Identity Keys, or AIKs, are used for routine transactions.

TPM manufacturers will provide the endorsement key pair and store this in tamper resistant non-volatile memory before shipping the TPM. A certificate, or *endorsement credential,* can then be created which contains the public EK and information about the security properties of the TPM. This endorsement credential should be signed by a certification authority, known as the TPME or Trusted Platform Module Entity, who can attest to the fact that the key contained in the certificate is a public EK whose corresponding private EK is stored in a TPM that conforms to the TCG standards. This TPME may be a third party or, if authorised to do so, it may be the manufacturer themselves.

Some organisations who wish to use the TPM may, for security reasons, prefer to use their own endorsement keys. To accommodate this, the standards allow the EK to be deleted and re-installed by the user. Of course if the user-generated endorsement credential is not signed by a TPME, its use may be limited.

The purpose of the endorsement credential is to prove that the corresponding private EK is stored in a genuine TPM. So, in keeping with policy to control exposure of the public EK, the private EK is *never used to generate signatures.* Thus, the public EK is never required to verify signatures so it does not have to be widely distributed. The public EK is only used for encrypting data sent to the TPM during the process of taking ownership and the process of creating AIK certificates. These processes are described in Sects. 7.3.12 and 7.4.4. Encrypting data with the public EK ensures that the plaintext can only be recovered by the particular TPM identified in the endorsement credential.

7.3.4 Attestation Identity Keys

As mentioned above the TPM Endorsement Key and Storage Root Key are stored in non-volatile memory. These keys never leave this secure storage during normal operation. A third type of key, the Attestation Identity Key (AIK), may also be stored within the TPM. This key may be regarded as an alias for the Endorsement Key. Each TPM can support many AIKs, thus the user can have many unlinkable keys that can be used to maintain anonymity between different service providers who require proof of identity. These AIKs must, therefore, be persistent and although they could be stored on the TPM non-volatile memory, for practical reasons the standards recommend keeping the AIK keys in secure external storage. The TPM however must provide a volatile storage area where one or more AIK keys can be loaded when in use.

7.3.5 Platform Configuration Registers

The Platform Configuration Registers (PCR) are unique features of the TPM architecture and are used to store *integrity metrics.* The integrity metrics stored in these registers measure the integrity of any code, from BIOS to applications, typically before the code is executed. Platform Configuration Registers may be implemented in volatile or non-volatile storage. However, these registers *must* be reset whenever the system loses power or restarts. If the registers were not reset then old integrity metrics might remain in the PCRs after a platform is rebooted and reconfigured. The standards specify that a TPM must have at least 16 Platform Configuration Registers and that each register stores 20 bytes. Registers 0 to 7 are reserved for exclusive use by the TPM, the remaining registers are free for use by the operating system and any application.

7.3.6 Programme Code

In common with smart cards, the TPM requires storage for the firmware that is used to initialise the device.

If the programme code is stored permanently on the tamper proof TPM, then it would be reasonable to assume that it is trustworthy. Thus there would be no need to check its integrity making this the obvious location to store the code that carries out the integrity checks on all other platform devices and code. That is to say, the programme code on the TPM is the obvious "root of trust" for integrity measurements described in Sect. 7.2. The TCG refer to such a root of trust as the CRTM, or Core Root of Trust for Measurement. Although the TPM programme code is the obvious choice for the CRTM, implementation decisions often require the CRTM be located in other firmware such as the BIOS boot block. Regardless of where the CRTM resides it should be considered as a *trusted component* of the system since if it fails all security policy based on integrity measurements will be broken.

7.3.7 Execution Engine

Like many smart cards, the TPM has an execution engine which runs the programme code described above. The execution engine responds to external commands by selecting the required programme code and executing it on the TPM.

7.3.8 Random Number Generator

Another unique feature of the TPM is the inclusion of a *true* random bit stream generator. Again the implementation is left to the developers so long as some random source is used, rather than a deterministic method. Having a true random bit generator is extremely valuable in any security application. In the TPM, random bit streams are used to seed a random number generator. The random numbers produced by this generator may then be used to construct keys for symmetric cryptographic applications. The random numbers may also be used to provide nonces and, by mixing with user input, to increase the entropy in pass phrases.

7.3.9 SHA-1 Engine

The SHA-1 message digest engine is an implementation of the Secure Hash Algorithm [16] SHA-1. This algorithm hashes the input data and produces a 20-byte digest. It also forms the basis of an HMAC [17, 18] (Hash-Based Message Authentication Code) engine, and is used in a number of cryptographic procedures carried out by the TPM, for example: in the computation of digital signatures and for creating key objects where a hash of the key may be required as part of an integrity protection mechanism.

7.3.10 RSA Key Generation

Generating keys suitable for use with the RSA algorithm [19] can be a computationally intensive task and since such keys are widely used in the TPM, for signing and providing secure storage, the standard specifies that the TPM should include a module specifically for this task [5]. The standard requires that a TPM is able to support keys *up to* a 2048 bit modulus. Moreover, there are certain keys used with the TPM that must have *at least* a 2048 bit modulus.

In other words, all implementations of the TPM are required to support up to 2048 bit RSA. Some keys are allowed to have a smaller modulus than this, depending on what they are used for. However there are certain keys that must have a 2048 bit modulus—or greater. Of course if the modulus is greater than 2048 bits there is no guarantee that all implementations of the TPM will support this since the only requirement is that the TPM is able to support keys with up to a 2048 bit modulus.

7.3.11 RSA Engine

Just as the generation of RSA keys is computationally complex, so is the execution of the algorithm itself. Therefore, the standards also require the TPM to have a dedicated RSA engine used to execute the RSA algorithm. The RSA algorithm is used for signing, encryption and decryption. Maintaining the principle of key separation, the TPM uses dedicated signing keys for signing data, and separate storage key pairs for encryption and decryption. It is worth noting that the TPM does not mandate the use of any symmetric crypto-engines. According to the TCG [5], "The TCG committee anticipates TPM modules containing an RSA engine will not be subject to import/export restrictions."

7.3.12 Opt-In

The Opt-In component and the concept of ownership, represent one of the biggest differences between smart cards and the TPM. Smart cards are in general, owned and customised by the Issuer before the consumer receives the device. The TCG, however, conscious of the perception that TPM enabled platforms will somehow be controlled by large remote organisations, have been very careful to provide mechanisms to ensure that it is *the user* who takes ownership and configures the TPM. The TCG policy is that the TPM should be shipped "in the state that the customer desires" [5]. Thus, it is up to the user to opt-in to use the TPM. Users are not compelled to use trusted computing, they only opt-in if they choose to do so by taking ownership of the device.

During the process of taking ownership, the TPM will make transitions through a number of states depending on the initial state in which the device was shipped. The state in which the TPM exists is determined by a number of persistent and volatile flags. Changing the state of these flags requires authorisation by the TPM owner, if he exists, or demonstration of physical presence. Proving ownership is discussed below, but the means of proving physical presence is determined by the platform manufacturer. The key point is that no *remote* entity, other than the TPM owner, should be able to change the TPM state [10].

The function of the Opt-In component is to provide mechanisms and protection to maintain the TPM state via the state of these flags.

7.3.12.1 TPM Operational States

There are several mutually exclusive states in which the TPM can exist, ranging from disabled and deactivated through to fully enabled and ready for an owner to take possession [5]. In the *disabled* state, the TPM restricts all operations except the ability to report TPM capabilities and to accept updates to the PCRs. Once a TPM is *enabled*, all its features are made available, provided ownership has been established. If an owner has not already been established, then the transition from disabled to enabled requires proof of physical presence.

A second state variable indicates whether the device is *activated* or *deactivated.* The deactivated state is similar to the disabled state. The difference is that when deactivated, a TPM may still switch between different operational states, for example to change owner or to activate. Once activated, all features are available [5].

A third state variable relates to ownership and determines whether the TPM is *owned* or *unowned*. For a TPM to be owned by a user, it must have an endorsement key pair, and a secret *owner authorisation data* known by the owner. Once in the owned state, the owner of the TPM may perform all operations including operational state change. The TPM needs to have an owner and be enabled for all functions to be available [10].

Depending on the configuration, cases may arise where different states have overlapping influence. As explained in the TCG architectural overview specification [5], in such situations, where TPM commands are available by one mode and unavailable by another mode, precedence is given to making the command unavailable.

7.3.12.2 Taking Ownership

When a user takes ownership of a TPM they establish a shared secret, referred to as *owner authorisation data,* and insert this data into secure storage on the TPM. This is, in effect, the owner's password and being able to demonstrate knowledge of this secret provides proof of ownership. Once owned, the TPM can control access to certain protected operations by requiring proof of ownership which can be demonstrated by entering the *owner authorisation data* to authenticate the owner.

The process of taking ownership requires that the owner authorisation data is protected from eavesdropping or theft by a malicious third party. This is one case where the endorsement credential is used, since the EK is the only key an unowned TPM has. It is the EK that establishes the secure channel to transmit the authorisation data to a genuine TPM. So, during the process of taking ownership, the owner requests the endorsement credential and after verifying this credential, retrieves the public EK and uses this to encrypt the shared secret. Only the TPM identified in the endorsement credential has access to the private EK, therefore the shared secret is only made available to the intended TPM. The process of taking ownership is completed by the creation of the Storage Root Key (SRK) which is generated by the TPM and stored in non-volatile memory. The SRK forms the root of a key hierarchy used to provide secure storage and, as with the private EK, the SRK *never leave*s the TPM.

7.3.13 Other Features

The TPM provides several other useful security features that are not always available in smart cards. One of these is the provision of monotonic counters which can, for example, provide a secure mechanism to prevent replay attacks. A second feature is the provision of time-stamping, although it is important to note that no absolute measure of time is possible, only measurement of time intervals. Finally, the TPM should provides mechanisms to create and manage audit trails

7.4 TPM Services

To conclude this section, we present an overview of the TPM operation to illustrate the basic security services provided. The foundation for the provision of these services is the concept of a "root of trust" from which other services such as authenticated boot, secure storage and attestation can be constructed.

7.4.1 Roots of Trust

As defined in Sect. 7.2, a trusted platform has at its foundation some trusted component. Since all other trust is built upon this foundation, it is known as a *root of trust*. An analogy may be made with root certification authorities (CA) in a Public Key Infrastructure (PKI). Although users may not trust the public key presented in a certificate, or even the authority who signed the certificate, so long as there is someone at the top of the certificate chain whom the user trusts, then it is reasonable to trust the key—assuming of course that all signatures are verified. The entity at the top of the certificate chain is often referred to as the *root CA* and his certificate referred to as a *root certificate*. In most web browsers, many root certificates are installed with the application and trusted implicitly by the users. Failure of this component, the root certificate, breaks the security policy of the PKI.

In the trusted platform architecture defined by the TCG, there are three distinct roots of trust: a Root of Trust for Measurement (RTM), a Root of Trust for Storage (RTS) and a Root of Trust for Reporting (RTR).

The Root of Trust for Measurement (RTM) must be trusted to generate integrity measurements for the processes that are running on the platform. This should ideally be a tamper proof component that boots very early in the boot process and therefore is able to measure all other components that are loaded after it. The TPM is an ideal candidate for the CRTM, but in practice the CRTM is more likely to be found in the BIOS boot block. This has been discussed in Sects. 7.2 and 7.3.6.

The Root of Trust for Storage (RTS) is a trusted component that provides confidentiality and integrity protection. The RTS can then be trusted to store either data, e.g. PCRs, or keys such as the SRK that allow data to be securely stored in external locations. Finally, the Root of Trust for Reporting (RTR) is a trusted component that provides reports of any integrity measurements that may have been made—attesting to the platform configuration. Both the RTS an RTR are provided by the TPM.

7.4.2 Boot Process

Figure 7.2, taken from the TCG Architectural Overview [5], shows the system boot process. The BIOS boot block, or trusted Boot Block (TBB) as shown in the diagram, contains the CRTM, and is the first process to boot. The CRTM is a trusted component and its integrity is not measured by any external code, but it may perform a self-check of its own integrity. Although not illustrated in the diagram, the CRTM should measure the rest of the BIOS before loading it. Once the BIOS is loaded, it takes control and measures the integrity of the OS Loader as shown in step 1 of the diagram. Control then passes to the OS Loader in step 2, and the OS Loader measures the integrity of the operating system. This process continues until the applications are loaded and executed in step 6.

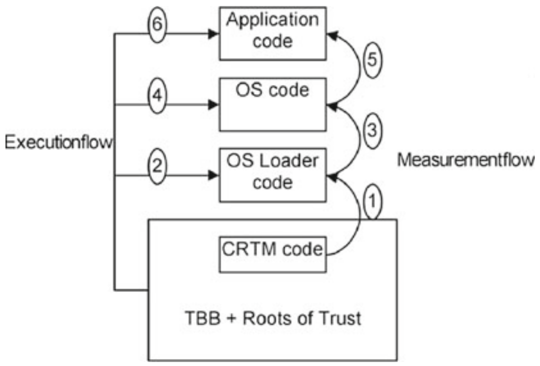


Fig. 7.2 Boot process [5]

With the introduction of Intel TXT in modern processors an alternative secure boot process may be used. This is sometimes referred to as "Late Launch" and is required the addition of two new CPU instructions. These instructions allow the CPU to be restarted in a safe manner and to execute digitally signed code during the reboot process to perform integrity measurements.

On the TPM, the integrity measurements at each stage are made by creating a SHA-1 digest of the code to be loaded. This digest is stored in one of the PCR registers , which are initialised to zero. The new integrity metric however does not simply overwrite the old PCR value. The process of updating (or extending) the PCR value concatenates the 20 bytes of data already held in the PCR with the 20 bytes of new data calculated by hashing the new code. These 40 bytes of data are then hashed again using the SHA-1 algorithm and the result written to the original PCR. In pseudo code: $PCR \leftarrow hash(PCR||hash(newcode))$. This way the PCR can store an unlimited number of measurements.

In order to interpret the value contained in the PCR, it is necessary to know the individual digests that have been added to it. These data are stored externally in what the TCG refers to as the Stored Measurement Log. Thus, if the data in the stored measurement log is known, and the PCR values are known, and trusted, then a challenger can verify the state of the platform.

7.4.3 *Secure Storage*

In Sect. 7.3.12 it was mentioned that the process of taking ownership resulted in the creation of a Storage Root Key, or SRK. This key is generated by the TPM and never leaves the device. It can only be accessed by demonstrating knowledge of a shared secret, in this case the *SRK authorisation data.* This shared secret is similar to the *owner authorisation data* and loaded into the TPM at the same time, during the process of taking ownership. As with the *owner authorisation data,* the *SRK authorisation data* is encrypted by the endorsement key before being sent to the TPM.

The SRK forms the root of a key hierarchy as illustrated in Fig. 7.3 which has also been taken from the TCG Architecture Overview [5]. This key hierarchy allows data, or keys, to be encrypted such that they can only be decrypted by accessing the TPM. In the diagram, the opaque data is encrypted under a specific storage key. The storage key is also encrypted and stored external to the TPM. To access this data the encrypted storage key is loaded into the TPM, via the key cache manager, and deciphered by the storage root key. Since the SRK never leaves the TPM, and the TPM is physically bound to the trusted platform, the opaque data can only be deciphered on that platform.

The TPM provides two mechanisms for secure storage: binding and sealing. The *binding* operation encrypts the data using a key that is managed by a particular TPM as described above. The *sealing* process adds to this by only allowing the deciphering process to proceed if the platform is in a specific configuration. This configuration is determined by data held in the PCR registers. Thus, when data is sealed, not only must the same platform be used to unseal the data, but that platform must be in a predetermined configuration before the data can be recovered.

7.4.4 Attestation

The AIK keys shown in Fig. 7.3 are the Attestation Identity Keys mentioned in Sect. 7.3.1. The purpose of an AIK is to provide user privacy when communicating with different sources. Although EK could be used to secure communications with different sources, since the EK is unique to the TPM, this could potentially allow the platform's identity to be linked between each source it chose to communicate with. The idea of the AIK is to provide a unique unlinkable identity for the TPM, for use with each different source. In each case, the AIK acts as an alias for the EK. The AIK credential is a certificate containing the AIK public key which proves that the corresponding private key is bound to a genuine TPM. This proof is guaranteed by a signature on the credential created by a trusted third party known as a "privacy CA".

To obtain an AIK a request is sent to the privacy CA together with the endorsement credential. This the second case where the public EK is exposed. The endorsement credential proves to the privacy CA that the request came from a genuine TPM. In response to the request, the privacy CA creates and signs the AIK credential, and encrypts this under the public EK contained in the endorsement credential. Thus the AIK is cryptographically bound to the TPM that contains the private EK. Users can create as many AIK keys as they wish and provided the privacy CA is trustworthy, these keys will remain unlinkable and provide the privacy required when communicating with different sources.

The private AIK key is managed by the TPM as illustrated in Fig. 7.3 and may be freely used to generate signatures. In particular, the private AIK can sign the contents of the PCR registers. This is how the TPM can be used to attest to the platform configuration. A relying party in possession of a public AIK can now challenge the trusted platform, and the attestation mechanism can be used to provide an integrity report describing the platform state.

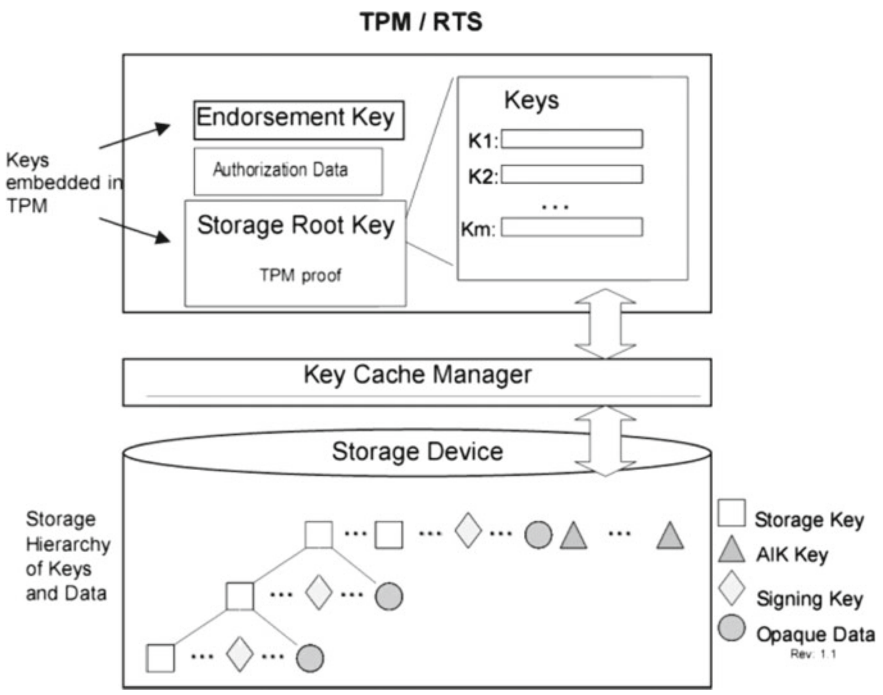


Fig. 7.3 Secure storage [5]

7.5 TPM 2.0

It is clear that TPM 1.2 relies heavily on the SHA-1 hashing algorithm. In 2005, Wang et al. presented an attack on SHA-1 which meant that collisions could be found easier than the theoretical limit would suggest [20]. This attack did not apply to the way the algorithm was used in TPM 1.2 [21]. However, prompted by this development, and in order to mitigate further developments in cryptanalysis, the TCG began a major review of the specification which led to the release of version 2.0 which was announced in April 2014 and standardised by ISO/IEC in 2015. As with v1.2, the specification has three volumes on architecture, structures and commands [22–24]; and a fourth volume on supporting routines [25]. The v2.0 specifications are written so that they can be processed by an automatic parser and have been used to generate a TPM simulator.

In addition to dealing with the problematic hash function, the main changes and enhancements compared to TPM 1.2 include

- Support for additional cryptographic algorithms
- Enhancements to the availability of the TPM to applications
- Enhanced authorisation mechanisms

- Simplified TPM management
- Additional capabilities to enhance the security of platform services

The following describes some of the changes that are pertinent to the discussion of the TPM and Smart Cards.

7.5.1 Capabilities

The Root of Trust of Measurement, Root of Trust of Storage and Root of Trust of Reporting (Attestation) are retained but mechanisms have changed for TPM 2.0.

Both versions of the specification have a Random Number Generator (RNG) but the TPM 2.0 specification states that this function should "meet the certification requirements of the intended market" [22].

Key derivation and storage capabilities are preserved but mechanisms have changed for TPM 2.0.

Both standards have protection against password guessing brute force attacks.

Non-Volatile Random Access Memory (NVRAM) is present but the TPM 2.0 specification allows more flexible use of it.

7.5.2 Comparison of Architectures

TPM 2.0 is designed to be more agile in terms of the cryptographic primitives it supports. To this end, TPM 2.0 supports RSA encryption and signature, ECC (Elliptic Curve Cryptography) encryption and signature, ECC-DAA (Elliptic Curve—Direct Anonymous Attestation) and ECDH (Elliptic Curve Diffie Hellman). Additional asymmetric cryptography algorithms can be certified by TCG and added to the specification. TPM 1.2 only supports RSA encryption and signature and RSA-DAA (RSA—Direct Anonymous Attestation). Moreover, TPM 2.0 supports symmetric cryptography including AES (Advanced Encryption Standard). In terms of cryptographic hashing, TPM 2.0 supports SHA-1, SHA-2 and SHA-256 and SHA-384 while TPM 1.2 is limited to SHA-1 and HMAC. Furthermore, TPM 2.0 supports key revocation via the key derivation function. If a key and its associated hierarchy have to be removed, then the only thing that needs to be done is to destroy the template and the secret entropy that generated this key.

Aside from the additional cryptography support, TPM 2.0 has enhanced authorisation which is significantly more useful than the authorisation function of TPM 1.2. In TPM 2.0, authorisation policies can be combined using logical rules. This allows, for example, multi-factor authentication.

As the use of the TPM has become more widely understood, the need for enforcing an opt-in policy has diminished. So, in order to make the TPM more user-friendly, TPM 2.0 is always on. Users can decide to use its full suite of functions or limit the usage to certain functions such as secure storage, cryptography engine and RNG.

Use of Non-Volatile RAM in TPM 1.2 was constrained to storing keys and was not customisable. TPM 2.0 allows users to specify how the NVRAM is configured for use as PCRs and secure storage. Keys in TPM 2.0 can be managed based on three separate key hierarchies: platform, storage and endorsement. TPM 1.2 only has a storage key hierarchy.

7.6 In Conclusion

This chapter has reviewed the structure of the TPM and identified several unique features that distinguish it from a typical smart card. One of the main differences between the two is the concept of ownership. Where smart card cards are usually owned and customised by the Issuer, the TCG have been careful to ensure that the TPM is owned by the user.

The TPM is designed to provide a number of security services to a platform which differ from those services provided by smart cards. These services include secure storage that can be used to protect data in a hostile environment and, when used with integrity reporting, can ensure that this data is only accessed in a controlled manner. The TPM also provides the mechanisms to implement authenticated boot processes. Perhaps the most unique feature of the TPM is the attestation mechanism. This allows challengers to verify the state of the platform, and which applications are running. It remains, however, the challenger's responsibility based on this attestation, to decide whether or not the platform is actually trustworthy.

References

1. Dept. of Defense, "DoD 5200.28-STD: Department of defense trusted computer system evaluation criteria", tech. rep., Department of Defense, 1985.
2. C. Mitchell, ed., Trusted Computing. London, UK: IEE Press, 2005.
3. E. Gallery and C. J. Mitchell, "Trusted mobile platforms", in FOSAD '07, International School on Foundations of Security Analysis and Design, vol. 4677 of LNCS, (Bertinoro, Italy), Springer-Verlag, Sep 2007.
4. S. W. Smith, Trusted Computing Platforms: Design and Applications. Springer-Verlag, 2005.
5. TCG, "TCG Specification Architecture Overview", TCG Specification Revision 1.4, The Trusted Computing Group, Portland, OR, USA, Aug 2007.
6. S. Pearson, "Trusted computing platforms, the next security solution", Technical Report HPL-2002-221, Hewlett-Packard Laboratories, Nov 2002.
7. B. Balacheff, L. Chen, S. Pearson, D. Plaquin, and G. Proudler, *Trusted Computing Platforms: TCPA Technology in Context.* Prentice Hall, 2003.
8. R. Anderson, "Cryptography and competition policy: issues with 'trusted computing' ", in PODC '03: Proceedings of the twenty-second annual symposium on Principles of distributed computing, (New York, NY, USA), pp. 3V10, ACM Press, 2003.
9. R. J. Anderson, Security Engineering - a Guide to Building Dependable Distributed Systems. Wiley, 2001.
10. TCG, "TPM Main, Part 1 Design Principles", TCG Specification Version 1.2 Revision 116, The Trusted Computing Group, Portland, OR, USA, March 2011.

11. S. Pearson. (ed.): "Trusted Computing Platforms: TCPA Technology in Context", Prentice Hall, 2003.
12. TCG, "TPM Main, Part 2 TPM Structures", TCG Specification Version 1.2 Revision 116, The Trusted Computing Group, Portland, OR, USA, March 2011.
13. TCG, "TPM Main, Part 3 Commands", TCG Specification Version 1.2 Revision 116, The Trusted Computing Group, Portland, OR, USA, March 2011.
14. E. Gallery, "An overview of trusted computing technology", in Trusted Computing (C. J. Mitchell, ed.), IEE, 2005.
15. S. Kinney, Trusted Platform Module Basics: Using TPM in Embedded Systems. Elsevier, 2006.
16. FIPS, "Specifications for the Secure Hash Standard", Tech. Rep. 180-2, Federal Information Processing Standards, National Technical Information Service (NTIS), 5285 Port Royal Road, Springfield, VA 22161, Aug 2002.
17. H. Krawczyk, M. Bellare, and R. Canetti, "HMAC: Keyed-hashing for message authentication", RFC 2104, IETF, 1997.
18. M. Bellare, R. Canetti, and H. Krawczyk, "Keying Hash Functions for Message Authentication", in Advances in Cryptology - Crypto '96 (N. Koblitz, ed.), vol. 1109 of Lecture Notes in Computer Science, Springer-Verlag, 1996.
19. R. L. Rivest, A. Shamir, and L. M. Adelman, "A method for obtaining digital signatures and public-key cryptosystems", Communications of the ACM, vol. 21, pp. 120V126, 1978.
20. X. Wang, L. Y. Yiqun, and H. Yu, "Finding collisions in the full SHA-1." Annual International Cryptology Conference, Springer Berlin Heidelberg, 2005.
21. W. Arthur and D. Challener, *A Practical Guide to TPM 2.0* ApressOpen, 2015.
22. TCG, "Trusted Platform Module Library, Part 1: Architecture", TCG Specification Version 2.0 Revision 00.99, The Trusted Computing Group, Portland, OR, USA, October 2013.
23. TCG, "Trusted Platform Module Library, Part 2: Structures", TCG Specification Version 2.0 Revision 01.16, The Trusted Computing Group, Portland, OR, USA, October 2014.
24. TCG, "Trusted Platform Module Library, Part 3: Commands", TCG Specification Version 2.0 Revision 01.16, The Trusted Computing Group, Portland, OR, USA, October 2014.
25. TCG, "Trusted Platform Module Library, Part 4: Supporting Routines", TCG Specification Version 2.0 Revision 01.16, The Trusted Computing Group, Portland, OR, USA, October 2014.

Chapter 8
Common Criteria: Origins and Overview

John Tierney and Tony Boswell

Abstract This chapter will consider how the Common Criteria for Information Technology Security Evaluation evolved, how they are defined and how they are used in practice. As an example we will look at how Common Criteria is applied to smart card evaluations. This chapter will not attempt to describe the full detail of Common Criteria, but will explore the scope of the criteria, the infrastructure that supports their use, and how protection Profiles and Security Targets are created to act as baselines for evaluations. As such it acts as an introduction to the use of Common Criteria, on which a reader can base further reading and practice in order to apply Common Criteria to real-world situations.

Keywords Smart card · Common Criteria · Security evaluation

8.1 Introduction

Security doesn't come for free: it costs time and it costs money. In a world where new IT security threats are identified on an almost daily basis we have moved from asking "Do I need security?" to the more difficult question of "What security should I have?" And the answer to this question is almost always "It depends". In order to determine the security measures I should look for in my IT products and systems, I need first to perform an analysis of my risks and balance their potential impacts against the costs of implementing countermeasures. Such a risk analysis is itself a difficult task, and relies on progressively greater levels of expertise as we drill down to the level of individual IT products and their secure operation.

If we take on the task of deploying and operating an IT system then in most cases we will be buying (rather than building) products that we connect together to create

J. Tierney
MasterCard, New York, USA
e-mail: john.tierney@mastercard.com

T. Boswell (✉)
DNV GL, Cambridge, UK
e-mail: tony.boswell@dnvgl.com

K. Mayes and K. Markantonakis (eds.), *Smart Cards, Tokens, Security and Applications*, DOI 10.1007/978-3-319-50500-8_8

the IT system, and at least some of these products will need to implement security countermeasures according to their role in the system. When it comes to working out what security features to look for in a product, and whether the advertised security features of a product work correctly, we face a difficult problem unless we are all to develop our own deep knowledge of each product and the nature of the security properties we require from it. Furthermore, we typically want more than to read about a product in its pre-sale documentation: we want to confirm properties by testing, and in some areas by examining design information about the product. Finally, we need to work out how to compare the solutions offered by different products, which will often describe their approach to security in different terminologies, related to different features and implementations: it would be a huge benefit to be able to read about security claims in a language and format that is common across all the products we are considering.

A common framework for expressing security requirements and the method of evaluating them would therefore help both users of the products (i.e. those who design systems based on the products), and the product developers (who would otherwise have to support each of their potential customers in carrying out their own evaluation tasks). Using such a framework, the risk owners and end users of systems should benefit from more clarity in the expression of requirements, assessment of product suitability, and confidence that the implementation of security properties can be confirmed. Common Criteria provides this framework: an internationally standardised approach to specifying security requirements and claims, and then evaluating and certifying a product against the claims that it makes.

This chapter will consider how the Common Criteria for Information Technology Security Evaluation evolved, how they are defined and how they are used in practice. As an example we will look at how Common Criteria is applied to smart card evaluations. This chapter will not attempt to describe the full detail of Common Criteria, but will explore the scope of the criteria, the infrastructure that supports their use, and how Protection Profiles and Security Targets are created to act as baselines for evaluations. As such it acts as an introduction to the use of Common Criteria, on which a reader can base further reading and practice in order to apply Common Criteria to real-world situations.

8.2 Evolution of National and International Standards

Publicly available standards for security evaluations began with the TCSEC—Trusted Computer System Evaluation Criteria,[1] which were originally published in the US in 1983. These provided a methodology for assessing security and defined seven sets of functional security requirements, ranging from D (Minimal Protection) to

[1] Also known as "The Orange Book", and one of the "Rainbow Books": a series of security standards and guidance documents published by the US National Computer Security Center in a range of coloured covers.

A1 (Verified Design). In the late 1980 s the UK Government introduced a set of UK Confidence Levels, ranging from L1 (lowest) to L6 (highest). Both US and UK schemes were essentially aimed at the government market and not at commercial products. In the early 1990 s a European Initiative combined the UK scheme with similar criteria in other European nations to form the ITSEC (Information Technology Security Evaluation Criteria). This was a European standard with the UK, Germany, France and the Netherlands participating to develop a well-defined set of security evaluation criteria. For the first time, there was an international security evaluation standard available and security certification began to be seen as a potential marketing tool—hence creating an additional motivation (beyond the government market) for commercial products to be evaluated and certified.

The first version of ITSEC stimulated new Canadian and draft US criteria in 1993; however by then it was recognised that there was both a need and opportunity to define an international standard that could have global reach. Version 1.0 of Common Criteria was released in 1996, followed by version 2.0 in 1998. At the time of writing, the current version of Common Criteria is v3.1 revision 4 (see [1–3]), which was published in 2012.[2] Common Criteria became an ISO standard (15408) in 1999 (v2.1) although it also remains freely downloadable from the Common Criteria Portal website (www.commoncriteriaportal.org). The criteria are accompanied by the Common Evaluation Methodology (see [4])—also an ISO standard (18045)—which defines the generic activities and work units that an evaluator carries out when evaluating a product according to the assurance requirements that it claims. The difficulty of developing and agreeing an international standard should not be underestimated given the cautious nature of governments where national security is concerned. What Common Criteria gives us today is a framework for security evaluations of government and commercial products based on a common language and process. This in turn creates the foundation for international mutual recognition of product certifications, in which a single product certification can be recognised beyond the nation where the certification was carried out.

8.2.1 International Recognition

Although Common Criteria and the Common Evaluation Methodology provide a *framework*, recognition of product certificates between governments is determined in practice by participation in formal international agreements. Two such agreements are mentioned here: the global Common Criteria Recognition Arrangement (CCRA) and the European SOG-IS mutual recognition agreement.

The CCRA (see [5]) defines the scope and limits of international recognition of Common Criteria certificates between (at the time of writing) 25 nations, and also

[2]The *revisions* contain mostly minor updates and corrections to clarify the criteria and their interpretation. A description of the CC maintenance process is given at www.commoncriteriaportal.org/cc/maintenance.

the requirements that a nation needs to meet if it is to issue its own certificates in a way that will be recognised by the other participants. Current participants in the CCRA are:

- Australia*
- Austria
- Canada*
- Czech Republic
- Denmark
- Finland
- France*
- Germany*
- Greece
- Hungary
- India*
- Israel
- Italy*
- Japan*
- Malaysia*
- Netherlands*
- New Zealand*
- Norway*
- Pakistan
- Qatar
- Republic of Korea*
- Singapore
- Spain*
- Sweden*
- Turkey*
- UK*
- USA*

Those nations marked with an asterisk in the above list are Certificate Authorizing participants in the CCRA, meaning that they each have an evaluation scheme in place that carries out evaluations and issues certificates. The other nations are referred to as Certificate Consuming participants, and do not currently operate their own evaluation schemes, but recognise certificates produced by the Certificate Authorizing members. This provides a global scheme for certificate recognition between CCRA participants (under certain conditions relating to the source and type of requirements and assurance components used)[3] and a way for developers to evaluate and certify a product in one nation and have that certificate recognised in any of the other CCRA nations.

[3]See the text of the CCRA on the Common Criteria portal (www.commoncriteriaportal.org) for more details of the recognition constraints.

SOG-IS provides a European Mutual Recognition Agreement (MRA) for international certificate recognition between a smaller group of nations, but with recognition of higher assurance levels and with interpretation for specialised application areas known in the agreement as "IT-technical-domains", including a domain for "smart cards and similar devices".[4] The following nations are SOG-IS participants:

- Austria
- Finland
- France**
- Germany**
- Italy*
- Netherlands**
- Norway*
- Spain**
- Sweden*
- UK**

Again those nations providing a Certification scheme are marked with an asterisk; those who are qualified in the "smart cards and similar devices" technical domain are marked with a double asterisk.

8.2.2 The Need for Security Benchmarks

As we have seen earlier, initial evaluation criteria such as TCSEC and the UK Confidence Levels were primarily aimed at providing assurance for products supplied to government and defence organisations. But the need for security benchmarks extends beyond government: all organisations deploying IT systems need to create a security architecture based on well-implemented security functionality in IT products. They want to know:

- What security features does a product implement?
- Are they implemented properly?
- What confidence do I have that they cannot be subverted or fail?

To properly answer these questions, organisations need to be able to reliably establish the security of products they may need or want. Furthermore, organisations want to be able to compare the security offered by products, in terms of whether each alternative meets some sort of minimum security baseline and what security properties it may offer beyond that baseline.

The main obstacles that organisations face in doing this arise because security evaluation is highly technical and specialised, and because developers cannot reasonably afford to go through detailed security discussions with every potential customer. This

[4]See the text of the mutual recognition agreement on the SOG-IS website at www.sogis.org.

leads to the need for independent specialists who can carry out a single assessment against a benchmark, with the result being published so that it can be reused by all potential customers. To give those customers the ability to compare products requires that we have a common way of expressing benchmark security properties, and also a common understanding of how to generate the assurance that these properties have been implemented correctly.

Common Criteria is designed to meet a large part of these needs. It establishes a way to set target requirements using a standardised form in a Protection Profile, and then for a product to make security claims (which may include a claim of conformance to a Protection Profile) in the standardised form of a Security Target. It then provides a standardised methodology for evaluating those claims, and the conformance to Protection Profile requirements. Finally, it provides a standardised infrastructure and practice for operating Certification schemes, based on detailed mutual recognition agreements (CCRA and SOG-IS).

8.3 Evaluation Practicalities

From a developer perspective, there are a number of practical issues to consider before embarking on an evaluation. These include:

- An evaluation will cost money—the Evaluation Laboratory will charge a fee, and the Certification Body[5] may also charge.
- An evaluation will take time—especially at the higher assurance levels which can typically take more than a year. An assessment needs to be made as to when the certificate will be available relative to product launch and critical sales and marketing milestones. This can significantly affect the value of the certificate to the developer: in the worst case a product could theoretically be obsolete before a high-level assurance evaluation is completed.
- An evaluation will need internal resources—there will typically be at least one person dedicated to managing the evaluation process internally and there will be evaluation-specific documentation to produce. Responding to evaluator questions and supporting evaluator testing will generally require some level of support from technical staff. In addition, the development process must ensure that it provides all the required documentation and processes that meet the requirements of the chosen assurance components.
- The developer will have to liaise with at least the Evaluation Lab and Certification Body, and possibly other participants in the evaluation. These entities may be located in different time zones and may have language differences.

[5] A Certification Body (also sometimes known as a Validation Body, but usually abbreviated as "CB") is an entity operated or sponsored by a national Common Criteria scheme to oversee evaluations carried out in that national scheme and to carry out certification on the basis of the technical reports from its evaluation laboratories.

- Evaluations using some Protection Profiles and assurance levels will include one or more site audits for the developer (depending on the number of sites involved in the product development and manufacturing lifecycle). This is usually the case for smart card evaluations.
- The developer needs to consider whether to perform the evaluation in parallel with the development (potentially delaying the product release) or in sequence after the development is completed (raising the question of what to do if the evaluation leads to a need for product changes). Most organisations seem to find that it is most efficient to perform evaluations in parallel with development, defining a timeline that identifies each specific deliverable and the evaluation activities based on it. Evaluating deliverables as soon as they are complete gives the developer the best opportunity to correct any problems at minimum cost (since it is widely recognised that changes are more expensive the later in the development cycle they are made). It needs to be recognised that some evaluation activities carried out at different times will be linked: for example, design analysis may be affected by later findings when testing samples of the product (e.g. if testing reveals a discrepancy with the design documentation), and therefore later evaluation work can lead to changes in earlier deliverables. However, a well-planned parallel approach minimises this danger.

8.3.1 Evaluation Overview and Roles

A high-level view of a Common Criteria evaluation, and the main roles involved, is shown in Fig. 8.1 (terminology used in the figure is then explained in the text below).

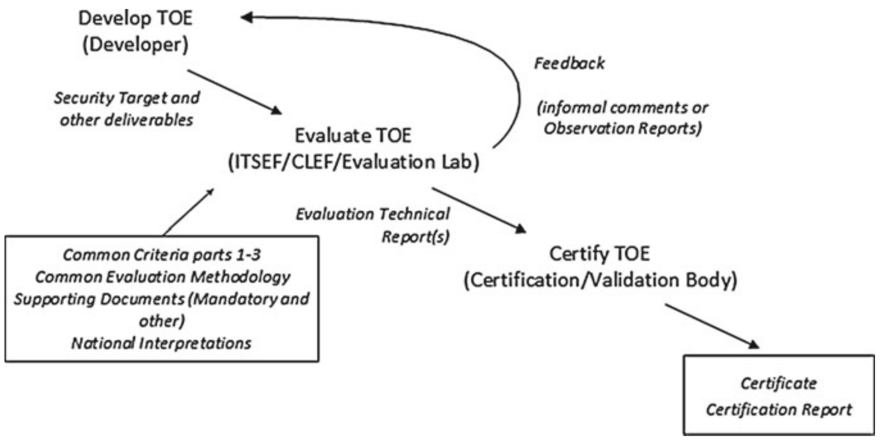

Develop TOE
(Developer)
Security Target and other deliverables
Evaluate TOE
(ITSEF/CLEF/Evaluation Lab)
Feedback
(informal comments or Observation Reports)
Common Criteria parts 1-3
Common Evaluation Methodology
Supporting Documents (Mandatory and other)
National Interpretations
Evaluation Technical Report(s)
Certify TOE
(Certification/Validation Body)
Certificate
Certification Report

Fig. 8.1 Evaluation process

The first point to note is that the product being evaluated is usually referred to in Common Criteria as the Target of Evaluation (TOE), which is defined in part 1 of the Common Criteria as a “set of software, firmware and/or hardware possibly accompanied by guidance”. The developer is responsible for delivering a Security Target that describes the TOE and its security claims, along with a range of other deliverables that give a more detailed description of the TOE (such as design and test information, and a copy of the TOE itself) to the evaluators. There is a separate role (the sponsor) defined in Common Criteria for the entity that instigates and pays for the evaluation, but often the developer will also take on the sponsor role.

The evaluators may be variously referred to as an ITSEF (Information Technology Security Evaluation Facility), a CLEF (Commercial Evaluation Facility) or an Evaluation Lab. The evaluators receive the deliverables and carry out the evaluation tasks defined in the Common Criteria and Common Evaluation Methodology. As a result of these activities they provide feedback to the developer, both informally and in formal Observation Reports (ORs), which may result in updates to the deliverables. On completion of the various evaluation activities, the evaluators produce one or more Evaluation Technical Reports (ETRs) which are sent to the Certification Body (CB—also sometimes known as a Validation Body). The CB then performs a number of oversight activities based on reviewing the ETR and ultimately (assuming the evaluation verdict is a “Pass”) issues the certificate and Certification Report (CR) for the product.

It is important to note that the Evaluation Lab is independent of the developer, and needs to be approved by a national Certification Body to carry out evaluations as part of that nation’s CC scheme. As well as achieving and maintaining accreditation to ISO 17025, all evaluation labs are subject to ongoing monitoring and oversight by the CB to confirm their capability and freedom from conflict of interest: this ensures that evaluation provides an independent, expert analysis of the product. The oversight function provided by the CB is particularly important because of the variety of specialised areas of knowledge involved in smart card evaluations, which span:

- IC hardware design and manufacturing
- Physics (of the underlying semiconductor technologies)
- Software development (application and operating system levels)
- Cryptography
- Statistics (e.g. for side channel attacks)
- Electronics (e.g. for designing and using side channel and fault induction test equipment)
- General IT and site security
- Product Development

The independence of the Evaluation Lab also brings a different perspective from that of its developer. Since the Evaluation Lab is often given access to significant amounts of the developer’s intellectual property, it is important that the developer and Evaluation Lab form a trust relationship, and that the two parties put in place a suitable non-disclosure agreement. The CCRA [5] includes requirements on each national scheme to ensure that sensitive information is handled appropriately within the scheme (including by the labs).

The content of Protection Profiles and Security Targets will be discussed in more detail in a later part of this chapter, but some introductory description is given here, to help the reader in understanding their role in the evaluation process. All evaluations require the developer to supply a Security Target (ST) that describes the TOE and the functional and assurance requirements that the TOE claims to meet, using a specific format and approach defined in the Common Criteria. The Security Target thus serves as the baseline for the evaluation activities. A Security Target may optionally claim conformance to a Protection Profile (PP), which is an implementation-independent specification of security requirements for a type of product: if the ST claims conformance to a PP then a lot of its content (including functional and assurance requirements) is drawn directly from the PP, with the ST adding product-specific implementation detail. Getting the ST right is critical for the success of an evaluation and is the focus of a significant amount of early work from both developer and Evaluation Lab.

In general, a product needs to take account of PP requirements from an early stage in its design. Retrofitting a product to a Protection Profile often causes problems either in meeting particular functional requirements, or in conforming to assurance requirements that affect the development process and types of design and test documentation produced.

8.3.2 Evaluation Assurance Levels

Common Criteria defines a variety of assurance components, which have historically[6] been used in the form of seven predefined packages called EALs (Evaluation Assurance Levels), with EAL1 being the lowest and EAL7 the highest. More details can be found in [3], including its Tables 1–8 which provide an overview of the various assurance requirements required to support each EAL. These tables illustrate the ways in which assurance requirements (and documented evidence, mainly from the developer) become more and more demanding as the EAL increases.

8.3.3 Augmentation of Assurance Levels

The EALs defined in CC part 3 [3] can be "augmented": this means that additional assurance components can be added in selected areas. The effect is that a partial—but not complete—enhancement to a higher EAL can thereby be stated. As an example, in smart card Protection Profiles EAL4 is often augmented by adding additional

[6]Use of the assurance levels in CC part 3 still continues, but in some areas the most recent use of CC has emphasised the definition of assurance in terms of the individual components rather than packaged levels.

assurance components to require (i) a justification that sufficient security measures are applied to the development (and manufacturing) environment (ALC_DVS.2), and (ii) that a higher level of attack potential is analysed during vulnerability analysis (AVA_VAN.5).[7] Augmented EALs were historically denoted simply by adding a "+" sign, so the above example would be denoted by "EAL4+". However, this means that the same EAL4+ label could denote many different specific augmentations, making it difficult to compare two products both claiming EAL4+. As a result, current practice is to describe the augmentation by listing the higher assurance components,[8] so in the example given here the assurance level would be stated as "EAL4 augmented by ALC_DVS.2 and AVA_VAN.5".

8.4 Making the Decision to Evaluate

Common Criteria provides a scheme under which developers can certify their products but, as noted in the "Evaluation Practicalities" section above, the developer must ensure they create a suitable business case. Evaluations cost time and money, especially when the higher EALs are used, and also require use of internal resources (some of which may be scarce design and test resources) to create the necessary deliverables and to support an evaluation. Any prudent commercial organisation will therefore want to assess the costs and benefits involved before undertaking an evaluation.

Issues to consider from a developer perspective include:

- Do the customers for this product mandate Common Criteria certification to a specified level or against a particular Protection Profile? (This is typically a requirement for supplying product to government or defence organisations, but can also apply to financial products).
- Is Common Criteria commonly used by competing products and/or partner products[9]? Will a Common Criteria certificate differentiate us from competing products in our target market sector? Or is certification a de facto requirement even if not formally mandated by the customers?
- Will developing a product to meet Common Criteria requirements, and undergoing an independent evaluation, help us to improve the quality of our development process and, ultimately, our product (e.g. because developing a product to Common Criteria standards will ensure that there is a rigorous focus on the design, implementation and testing of security functionality)?

[7]The topic of attack potential calculations (which basically involve deriving a number representing the difficulty of an attack) for actual and potential vulnerabilities is too big to discuss in this chapter. However, the interested reader is referred to [6] for details of how this is done for smart cards and related products.

[8]The order in which the components are listed is not significant.

[9]For example, if a software application requires certification then this will often imply a need or benefit for the underlying hardware also to be certified. See the discussion of composite evaluations later in this chapter.

Even Low Assurance Level (e.g. EAL1) evaluations will take several months to complete. Mid-level (e.g. EAL4) evaluations typically take between 6 and 9 months. High-level evaluations (EAL6, EAL7) take significantly longer. Any evaluation timescale is highly dependent on factors such as the need for rework on deliverables and the time taken to answer evaluator queries. Not surprisingly the time depends significantly on how well-prepared the developer is, and the first evaluation of a product usually takes longer than subsequent evaluations of new product versions (not only because there will be a set of previous deliverables for the developer to work from, but also because the developer will have experience of how to put in place the necessary evaluation support resources).

Costs will, again, vary dependent on the EAL required, but include:

- Direct fees to the Evaluation Lab and Certification Body
- Travel costs for meetings and site audit visits (where required)
- Internal resource costs for preparing deliverables and supporting the evaluators
- Any external resource costs (e.g. consultancy, contractors).

8.5 Developing Protection Profiles and Security Targets

Protection Profiles and Security Targets are quite similar in content: in fact a Protection Profile (PP) can be seen as an implementation-independent Security Target (ST), and a Security Target can be seen as a Protection Profile that has been mapped on to a specific product. The documents therefore have many sections in common (as can be seen in the description of contents in CC part 1 (see [1])).

A Protection Profile is a statement of security requirements for a type of product, such as the secure chip on which a smart card would be built, a smart card operating system, a smart card payment application, or a machine-readable travel document (i.e. the contactless chip that is now included in many current passports). It describes the security problem that the type of product is designed to solve, but remains independent of any implementation detail and is therefore not specific to any particular product. A PP is essentially a template that describes a generic security analysis (e.g. threats, assumptions, organisational security policies, and security objectives), leading to the definition of certain Security Functional Requirements based on Common Criteria part 2 (see [2]) and a set of assurance requirements from Common Criteria part 3 (see [3]).

PPs are intended to be useful for purchasers to use as a procurement requirement: this means, of course, that the purchaser must first examine the PP in some detail to check that it is suitable to express their requirements. Increasingly PPs are developed and controlled by "technical communities" that include participants from a variety of organisations including developers, evaluators, certifiers and risk owners (i.e. organisations that will use the TOE type to protect assets)—this is a way of ensuring that the resulting PP will be both useful and feasible for products to meet.[10] Claiming conformance to a Protection Profile in a Security Target is optional, but if a customer uses a PP as a procurement requirement then it becomes necessary in practice for a product to claim conformance in its ST.

A Security Target describes a similar security analysis, but does so in the context of a specific product (or family of similar products). An ST can be written standalone, without reference to a PP, in which case the ST gives a security analysis and states a set of security requirements that are determined by the product developer alone. But increasingly the preferred approach is for an ST to claim conformance to a PP that has been developed from a technical community, as noted above. There are a number of PPs defined and in common use for smart card products, and in general this chapter assumes that a PP will be used. Where an ST claims conformance to a PP it will therefore typically copy most of its security analysis sections (e.g. threats, assumptions, organisational security policies, and security objectives) from the PP. The ST author then adds product-specific introduction sections, and a TOE Summary Specification section that describes how the specific product implements the Security Functional Requirements from the Protection Profile.

This section will look at some details of how specific sections of these documents are developed.

8.5.1 The Security Problem Definition

The Security Problem Definition is the starting point for the security analysis in a PP or ST, and sets out the baseline from which security functional and assurance requirements will be derived, and against which their appropriateness will be judged. There are three main inputs to this definition:

- The purpose of the TOE
- The assets requiring protection
- The operational environment in which the TOE will be used

The **purpose** of the TOE describes the general information technology and application domain problem(s) that the TOE is intended to solve.

The **assets** requiring protection will generally be identifiable as things that an owner, and an attacker, place a value on (the value of the asset will often be the motivation for an attack). The assets will typically require protection of one or more of their confidentiality, integrity and authenticity (other properties might be relevant, but these are the most common). An initial asset identification will usually be based on the artefacts of the application domain in which the TOE is used, such as

[10]Indeed there are specific definitions and separate treatment of collaborative Protection Profiles (cPPs) developed from recognised international Technical Communities (iTCs) described in the CCRA (see [5], especially the definitions in Annex A and description of Collaborative Protection Profiles in Annex K).

monetary value represented as a record of an account balance held on a smart card. However, analysis of the ways in which these are represented and protected on a TOE may then lead to identification of other technology-based assets such as secret or private keys, passwords and PINs, or even configuration settings of the TOE (e.g. because a configuration setting may determine whether some other asset is stored in encrypted or unencrypted form, which users have access to some other asset, or whether authentication is required for access to another asset).

The **operational environment** of the TOE comprises not only the obvious physical environment in which the TOE is intended to operate, but more generally includes any other factors that represent conditions that the TOE can rely on. For example, the physical environment may consist of a server room with strong access controls in the case of a web application, or may be completely unconstrained as in the case of a typical smart card application. In the case of the smart card environment, the Security Problem Definition would need to assume that attackers have unrestricted access to the TOE and can spend as much time as they like attacking it, with any means at their disposal (therefore including use of sophisticated tools such as focussed ion beams to image and alter circuitry).

Having considered these inputs, the PP/ST author then defines the security problem that the TOE is intended to solve in terms of:

- **Threats** (defined in terms of an asset under attack, the attack method and the agent carrying out the attack)
- **Organisational Security Policies** (security requirements, such as choice of cryptographic algorithms and key lengths, typically determined by risk-owning organisations or regulatory bodies related to the assets that the TOE protects)

Assumptions (security-related features or conditions that the TOE needs to rely on in its operational environment). These form the inputs to the next phase of PP/ST creation, and are shown in the context of the PP/ST structure in Fig. 8.2.

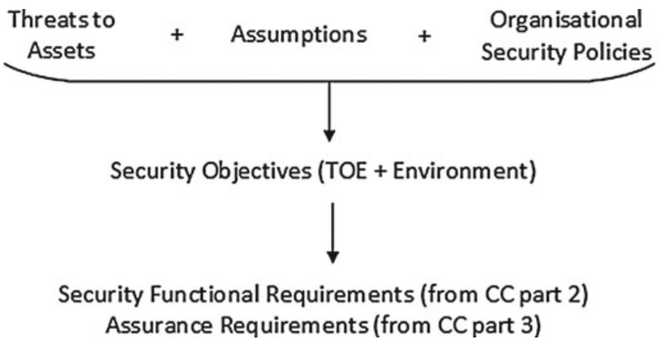

Threats to Assets
+
Assumptions
+
Organisational Security Policies
Security Objectives (TOE + Environment)
Security Functional Requirements (from CC part 2)
Assurance Requirements (from CC part 3)

Fig. 8.2 Protection Profile/Security Target security analysis structure

8.5.2 Security Objectives Definition

Using the elements of the Security Problem Definition as input, the PP/ST author now derives the security objectives for the TOE, and separately identifies the security objectives that the TOE relies on from its operational environment. The PP/ST author also writes a "security objectives rationale" to show that the security objectives for the TOE and its environment combine to adequately address the security problem, showing that the threats are countered (by a combination of TOE and operational environment), the organisational security policies implemented (again by a combination of TOE and operational environment), and the assumptions justified (typically by the presence of corresponding security objectives for the environment).

The Security Objectives for the TOE and its environment then form the inputs into the next phase.

8.5.3 Security Requirements Definition

The PP/ST author defines the TOE security requirements by translating the security objectives for the TOE (from the Security Objectives definition) into the form of the standardised Security Functional Requirements (SFRs) in CC part 2 (see [2]). In many cases this means selecting an appropriate combination of generic functional components directly from CC part 2 and then filling in certain "parameters" of the SFR according to the TOE type (in the case of a PP) or the product (in the case of an ST). Examples of this "completion" of an SFR are given in the example section below. Although the default approach is to select existing SFRs from CC part 2, it is also possible to define new SFRs in the style of CC part 2, to cover situations where a function is not present in the existing set, or would be difficult to express (and therefore potentially confusing to readers of the ST). Such SFRs are referred to as "extended" SFRs (and the resulting PP or ST is referred to as "Part 2 extended"). A typical example of an extended SFR is to define requirements on a random number generator (see, for example, FCS_RNG.1 in Sect. 5.1 of [7]).

The requirements are also compared to the TOE objectives in a "security requirements rationale". This follows a similar approach to the security objectives rationale, and demonstrates how each security objective corresponds to one or more SFRs, thus justifying the complete coverage of the security objectives.

As well as defining the SFRs, the PP/ST author also selects the required assurance components from CC part 3 (see [3]) at this point. This selection may involve choosing a complete assurance level from CC part 3, or the choice may be made at the level of individual assurance components (see the discussion of assurance components in the example below). It is also possible to define extended assurance components in the same way as for extended functional components, however this is rarely done (not least because in general such components are not in the default scope of the mutual recognition arrangements).

8.5.4 Create TOE Summary Specification

A Security Target will include a TOE Summary Specification that maps the SFRs to specific product security features, in order to explain how the SFRs are implemented in terms of the product's operating model and terminology. (Because a PP is product-independent there is no equivalent section in a PP.) This is intended to be a technical description that enables a reader (and potential customer) to understand the mechanisms that the TOE uses, but will not contain a developer's proprietary information because the ST is a public document.

8.6 An Example

As an example, we will look at a Protection Profile for a "security IC", which is used for many smart card chips (see [7]). This PP is used to certify chips that will then form a suitable base on which to place a CC-certified operating system and/or CC-certified application (the PP takes deliberate account of its use in later composite evaluations when software is added to the TOE: this is included in the product lifecycle described in Sect. 1.2.3 of the PP, and a further discussion of the role of the PP in a composite TOE is given in Sect. 1.2.5 of the PP). The examples here will not be exhaustive, and in particular will not cover the more recent use of collaborative Protection Profiles (cPPs): the reader is referred to the Common Criteria portal (www.commoncriteriaportal.org) for lists of, and links to, other Protection Profiles and Security Targets.

The TOE for this PP consists primarily of the hardware but, depending on the specific product, it may also include certain "IC Dedicated Software" that is supplied with the hardware: this covers software that is used during the manufacturing process (but which is not available after manufacturing) and the optional supply of support software for use by operating systems or applications (e.g. the IC manufacturer might supply cryptographic libraries that are designed to make optimum use of the hardware features). An ST that claims conformance to this PP will therefore identify not only a specific IC but the software that comprises its IC Dedicated Software (to the level of the particular versions of each software item).

It is also noted that the PP includes a large amount of material describing how to interpret CC and the PP requirements for the smart card environment (e.g. see Sects. 6.2.1 and 7.1–7.3 in [7]).

8.6.1 Example Security Problem Definition

The **purpose** of the TOE is to provide a secure hardware platform on which secure software can be built to provide applications to end users. This TOE type is perhaps unusual in that the security IC itself does not have a significant direct interaction with the end user: this will be the job of the software that is loaded later (and which will form part of a later composite TOE).

The **assets** are defined as:

- User data (requiring confidentiality and integrity protection)
- The embedded software (requiring confidentiality and integrity protection)
- The security services provided by the IC to the embedded software.

Note that the "embedded software" here is not the IC Dedicated Software supplied as part of the TOE: rather it is the software added to the TOE in order to provide the operating system and/or application. The point of both the user data and embedded software assets in this case is to establish that the task of the TOE is to provide general purpose mechanisms to protect whatever data and software a software developer happens to need for their operating system and/or application.

The **operational environment** of the TOE is generally uncontrolled: there may be some element of control by a software developer or card issuer when the chip is delivered from the hardware manufacturer and the card is prepared for issue to a particular cardholder, but once it has been released to the cardholder then no controls can be assumed. The general premise of the PP is therefore that attackers (who may in some cases be the cardholders themselves) have unrestricted access to the TOE and can apply whatever tools and whatever investment of time that they wish.

The **threats** to the assets are defined at a high level as:

- Manipulation of any data held by the TOE (including user data, but also considering embedded software as data)
- Manipulation of the TOE that might disable or weaken other protection measures
- Disclosure of data in an unauthorised way (including, for example, via side channels).

This leads to a definition of the formal list of PP threats in terms of the types of attack that the TOE will protect against:

- Physical probing (enabling an attacker to read data)
- Environmental stress leading to malfunction (e.g. applying unusual voltage or clock conditions)
- Physical tampering (in which the attacker modifies data on the TOE, or the behaviour of the TOE)
- "Leakage", meaning disclosure of data via inherent side channels (such as differential power analysis)
- "Forced Leakage", meaning disclosure of data by side channels that are created or enhanced by the attacker
- Abuse of functionality (this typically refers to attempts to reactivate the test mode functionality that is intended for use in manufacturing and which is disabled before the chip is released to end users)

- Poor quality random numbers (e.g. this could compromise the security of keys generated using the random number generator, or could enable an attacker to guess the nonces used by application software to protect against replay attacks in application software).

There is one **organisational security policy** defined in the PP. This places a requirement for unique identification of the TOE in each physical chip.

The **assumptions** are as follows:

- Protection of the TOE during later manufacturing processes: this passes on an obligation for the recipient of deliveries of the TOE from the manufacturer to protect it whilst performing the subsequent processes that are necessary to add/activate embedded software and to prepare the chip for delivery to a cardholder
- Secure use of the hardware features by embedded software: this passes on an obligation for embedded software to use the features of the TOE appropriately, in accordance with the guidance that accompanies it (this ensures that, for example, the side channel protection features are used correctly when performing cryptographic functions in application software)
- User data protection: this passes on an obligation for embedded software to use the TOE features in a way that will properly protect user data (this recognises that inappropriate embedded software design could itself put user data at risk, regardless of the protection in the chip hardware).

8.6.2 Example Security Objectives Definition

The TOE security objectivesdefined in the PP correspond closely to the threats and organisational security policy in the security problem definition:

- Protection against physical probing
- Protection against malfunctions
- Protection against physical manipulation
- Protection against inherent leakage
- Protection against forced leakage
- Protection against abuse of functionality
- Random number quality
- TOE identification.

Similarly, three security objectives for the operational environment are defined in the PP, with each corresponding to one of the assumptions.

The close correspondence of the objectives to the threats, organisational security policy and assumptions makes creating the security objectives rationale a relatively straightforward task. (For other types of TOE there might be a larger transformation step from the description of user-level threats to the mechanisms that a TOE uses to protect against them, and in such cases the security objectives rationale would be expected to be more complicated but also to add more value.)

8.6.3 Example Security Requirements Definition

The catalogue of Security Functional Requirements in Common Criteria part 2 provides a large set of SFRs grouped into the following classes:

- Audit
- Communication
- Cryptographic support
- User data protection
- Identification and authentication
- Security management
- Privacy
- Protection of the TSF (TOE security functions)
- Resource utilisation
- TOE access
- Trusted path/channels

Within each class, a number of families are defined, and within each family are the individual SFRs that can be chosen for use in PPs and STs. Some SFRs are defined to have dependencies on other SFRs: for example the SFR that defines a generic cryptographic operation (FCS_COP.1) has dependencies on other SFRs that deal with the origin of the key to be used in the operation (which in general will either be imported or generated in the TOE itself).

As an example of completing an SFR for a product type, consider the following SFR from the Protection of the TSF class (FPT_PHP.3) as it is found in CC part 2:

> FPT_PHP.3.1 The TSF shall resist [assignment: physical tampering scenarios] to the [assignment: list of TSF devices/elements] by responding automatically such that the SFRs are always enforced.

(Here TSF means TOE Security Functionality, which can be thought of as the combination of all the functionality that implements the set of SFRs in a PP or ST). The example shows two assignments in square brackets which are for completion by the PP or ST author.

Now consider the same SFR as it is used in Sect. 6.1 of [7]:

> FPT_PHP.3.1 The TSF shall resist physical manipulation and physical probing to the TSF by responding automatically such that the SFRs are always enforced.

Here the first assignment has been completed by the PP author to refer specifically to physical manipulation and physical probing, which are defined in more detail in other parts of the PP (the reader here can see, for example, that these relate directly to the language of the threats from the PP as discussed above). Similarly the second assignment has been completed to ensure protection of the whole TSF (the term "TSF" here is implicitly acting as a list of all TSF devices and elements).

In practice a chip might implement this SFR by methods such as an active shield layer to detect attempts to access its lower circuit layers and protect the IC by preventing sensitive data or processing from being carried out.

A more software-oriented example might be as follows:

FDP_SDI.2.1 The TSF shall monitor user data stored in containers controlled by the TSF for [assignment: integrity errors] on all objects, based on the following attributes: [assignment: user data attributes].
FDP_SDI.2.2 Upon detection of a data integrity error, the TSF shall [assignment: action to be taken].

This SFR might be completed in a PP or ST as follows (note that this is different to the more hardware-oriented use of FDP_SDI.1 in [7]):

FDP_SDI.2.1 The TSF shall monitor user data stored in containers controlled by the TSF for checksum errors on all objects, based on the following attributes: checksum errors on secret key data.
FDP_SDI.2.2 Upon detection of a data integrity error, the TSF shall raise an exception and reject all subsequent commands.

The security requirements rationale in the PP then takes each objective and maps each to a number of SFRs, including a text discussion in each case to give a fuller explanation of how the combined set of SFRs guarantees that all aspects of each security objective have been covered. The rationale makes clear how interdependencies arise between some of the objectives (e.g. the SFRs that implement protection against physical manipulation ensure that hardware countermeasures against leakage cannot be physically disabled).

The assurance requirements are also defined at this stage, and the example PP in this case begins by selecting the standard EAL4 level and then augments it by replacing ALC_DVS.1 with ALC_DVS.2 and AVA_VAN.2 with AVA_VAN.5. (The meaning of these particular augmentations was discussed in Sect. 8.3.3 above).

8.7 Assurance Components: Defining the Deliverables

The assurance components selected from CC part 3 set requirements on the information content of the evaluation deliverables given by the developer to the evaluator. Each of the assurance components is divided into three types of requirement:

- Developer action elements: these are the high-level things that the developer has to do and the items that the developer needs to deliver—for example, ADV_FSP.4 requires that "[t]he developer shall provide a functional specification" and that "[t]he developer shall provide a tracing from the functional specification to the SFRs".

- Content and presentation elements: these capture the required content of the deliverables, in terms of the information and sometimes its structure—for example, ADV_FSP.4 includes a requirement that "[t]he functional specification shall identify and describe all parameters associated with each [SFR-related interface]". Other design-related requirements in the ADV_TDS family require presentation of the TOE design at two hierarchic levels (referred to in CC as "subsystems" and "modules") and ADV_FSP.6 (used only in the EAL7 assurance level) requires the functional specification to use a formal (mathematical) notation.
- Evaluator action elements: these are the high-level activities that the evaluator carries out to assess the deliverables. Often (as in ADV_FSP.4) the main activity will be to "confirm that the information provided meets all requirements for content and presentation of evidence"; ADV_FSP.4 also adds "The evaluator shall determine that the functional specification is an accurate and complete instantiation of the SFRs". These very high-level actions are refined into more detailed work items in the CEM (see [4]).

Assurance requirement components are defined in CC part 3 for the following areas[11]:

- ADV class—development
  - Security architecture
  - Functional specification
  - TOE design
  - Implementation (e.g. hardware design language, chip layout plans, software source code)
  - Internals (i.e. requirements for well-structured design)
  - Security policy modelling
- AGD class—guidance documentation
  - Preparative procedures
  - Operational user guidance
- ALC class—life cycle support
  - Life cycle definition
  - Configuration management capabilities
  - Configuration management scope
  - Delivery
  - Development security (i.e. security of development environment)
  - Tools and techniques
  - Flaw remediation

[11] CC part 3 also defines assurance components for the evaluation of Protection Profiles and Security Targets, but these are not discussed here.

- ATE class—tests
  - Functional tests
  - Coverage (of functional testing)
  - Depth (of functional testing)
  - Independent testing (i.e. requirements on testing by the evaluator)
- AVA class—vulnerability assessment
  - Vulnerability analysis

If a predefined assurance level from CC part 3 is used then as the assurance level increases from 1 to 7 the number of assurance classes included increases, as does the scope and rigour demanded in certain areas of the requirement. It is obviously important for a developer to read the assurance components used in their PP/ST in order to prepare the right set of deliverables for the evaluator; however it can also be worthwhile for the developer to read the evaluator activities defined for these components in the CEM, since the explanation of how the evaluators will use the deliverables can help in understanding the necessary content.

8.8 Composite Evaluations

As was mentioned in Sect. 8.6 above, an IC will generally be the basis for a smart card product that in fact consists of a number of layered sub-products: typically an application running on an operating system, which in turn runs on the IC. It would therefore be beneficial to support the combination of certified parts in order to create a certified composite product. This is referred to in CC as a composite evaluation (and the TOE made up of separate parts is referred to as a composite TOE). Although the same idea of composition can be used with other TOE types, currently the most common use of this approach is in the smart card world, and specific CC supporting documents have been produced in order to deal with composite evaluations of smart card products (see [8]).

There are various reasons for evaluating the parts of a composite TOE separately, and one of the most important is that it separates the evaluation responsibilities between the developers of the different parts (since in general the developers will be different, each bringing specialist knowledge of an area to the design of their product). Thus, the IC developer takes responsibility only for the IC evaluation, the OS developer for the OS, and so forth. Taking this approach also enables the developer of one of the lower level products to reuse their product certification for multiple higher level products, without having to repeat the lower level evaluation each time.

However, this also raises one of the most difficult aspects of composite evaluation: dealing with the security-related interactions between parts of a composite TOE. For example: in general an IC with strong countermeasures against side channels can still result in a composite product that exhibits significant side channel vulnerabilities if the software either fails to use the countermeasures properly, or is written in a way that results in such strong leakage that the hardware countermeasures alone cannot protect it. Similarly, since hardware cannot usually deal with all possibilities for fault induction attacks, software will generally need to deal with at least some of the consequences of faults deliberately induced by an attacker in the hardware. This means that it is important for some of the results of the evaluation of a lower part of the composite TOE to be made available to the evaluator of the next level (e.g. the OS evaluator needs to know something about the results from the IC evaluation).

A mechanism for providing this sort of transfer has been included in the supporting document for smart card composite evaluations in [8]. This requires the composite evaluation to carry out certain specific composition-related activities based on information provided in an "ETR for Composition" which is produced as an output from the lower level TOE for use in the composite evaluation. These activities start at the ST level to check that dependencies of the higher level product have been included in the evaluation scope for the lower level TOE, then cover process activities such as delivery to ensure that the lower level TOE is handled in accordance with its assumptions, and finally include appropriate testing of potential interactions and dependencies between the different products. This includes, for example, a check that relevant guidance for the use of an IC has been followed in the creation of an OS. It also enables the evaluator of the composite TOE to include relevant hardware-related tests that focus on their impact on the composite TOE—for example, laser fault induction attacks may be used to efficiently target critical decision points in the OS software based on results of similar tests in the original IC evaluation. The intention is thus to enable interactions between the parts of a composite TOE to be sufficiently analysed without having to repeat the lower level evaluation(s) for each higher level TOE, and without the evaluator of the composite TOE having to negotiate access to the deliverables (e.g. design documentation) for the lower level TOE(s).

The composite evaluations can therefore be seen to have an implicit sequence: the results of the IC evaluation are used in the OS evaluation and therefore the IC evaluation must complete before the OS evaluation. The same reasoning applies to an application running on an OS of course. This still allows the possibility for large parts of the separate evaluations to run in parallel, but the final test stages will need to follow a strict order. It is possible that the separate evaluations will be done not only by different evaluators in different evaluation labs, but also in different schemes based in different countries. This emphasises the importance of mutual recognition between the different nations, and also reminds us that Certification Bodies are likely to need to communicate at the certification stage of a composite evaluation, at least for the CB certifying the composite TOE to confirm the certified status of the lower level components.

8.9 In Conclusion

Common Criteria is an international standard with strong and evolving international mutual recognition arrangements. CC is based on certification by national bodies supervising independent evaluation laboratories that call on a wide range of specialist technical skills and experience to perform security evaluations in a rigorous and well-defined manner.

Evaluations can be used as marketing tools, but are most useful when they are based on the security requirements of a Protection Profile produced by a community that includes risk owners and represents the security concerns of the community from developers to end users. Certification in this context gives confidence that developers have focussed the development process on ensuring that appropriate product security features are correctly implemented and tested.

Successful evaluations can be crucial tools in meeting customer requirements (and therefore in supporting product sales for a developer), but they need careful preparation and management (possibly including interactions with other evaluations for composite TOEs) in order to manage the cost and timescale parts of the business case.

Finally it is important to note that Common Criteria evolves, both as a standard and in terms of the infrastructure surrounding it. New supporting documents are produced to capture interpretations and best practice in particular areas such as smart cards, and there is an increasing focus on improving the harmonisation of technology-specific aspects between evaluations, evaluation laboratories, and certification bodies. This is reflected in the evolution of mutual recognition arrangements and their supporting processes and documents. In this way, Common Criteria intends to reposition itself as required to respond to a changing picture of both threats and the role of IT in the world to come.

This paper is the result of the author's experience of security evaluations and represents the author's personal views only and not necessarily those of DNV GL or any of its affiliates.

8.10 Useful Websites

The reader may find the following websites useful:

- www.commoncriterialportal.org—the main site for Common Criteria (and the best place to start): this provides links to CC documents, national schemes, evaluation laboratories, and certified products
- www.ncsc.gov.uk—the UK NCSC site (the Common Criteria pages can be found under the "Marketplace" heading)
- www.sogis.org—the site for the SOGIS mutual recognition agreement.

8.11 Glossary

CB	Certification Body
CC	Common Criteria
CCRA	Common Criteria Recognition Agreement
EAL	Evaluation Assurance Level
IC	Integrated Circuit
ISO	International Standardisation Organisation
IT	Information Technology
ITSEC	Information Technology Security Evaluation Criteria
MRA	Mutual Recognition Agreement
OS	Operating System
OSP	Organisational Security Policy
PIN	Personal Identification Number
PP	Protection Profile
SFR	Security Functional Requirement
SOGIS	Senior Officials Group Information Systems Security
ST	Security Target
TCSEC	Trusted Computing Security Evaluation Criteria
TOE	Target Of Evaluation
TSF	TOE Security Functionality

References

1. Common Criteria for Information Technology Security Evaluation - Part 1: Introduction and general model, Version 3.1 Revision 4, September 2012, CCMB-2012-09-001 (available from the 'Publications' section at www.commoncriteriaportal.org)
2. Common Criteria for Information Technology Security Evaluation - Part 2: Security functional components, Version 3.1 Revision 4, September 2012, CCMB-2012-09-002 (available from the 'Publications' section at www.commoncriteriaportal.org)
3. Common Criteria for Information Technology Security Evaluation - Part 3 Security assurance components, Version 3.1 Revision 4, September 2012, CCMB-2012-09-003 (available from the 'Publications' section at www.commoncriteriaportal.org)
4. Common Methodology for Information Technology Security Evaluation - Evaluation methodology, Version 3.1 Revision 4, September 2012, CCMB-2012-09-004 (available from the 'Publications' section at www.commoncriteriaportal.org)
5. Arrangement on the Recognition of Common Criteria Certificates In the field of Information Technology Security, 2 July 2014 (available from the 'About the CC' section at www.commoncriteriaportal.org)
6. Application of Attack Potential to Smartcards, v2.9, May 2013, CCDB-2013-05-002 (available from the 'Publications' section at www.commoncriteriaportal.org)
7. Security IC Platform Protection Profile, version 1.0, 15 June 2007, BSI-PP-0035 (available from the 'Protection Profiles' section at www.commoncriteriaportal.org)
8. Composite Product Evaluation for Smart Cards and Similar Devices, v1.2, April 2012, CCDB-2012-04-001 (available from the 'Publications' section at www.commoncriteriaportal.org)

Chapter 9
Smart Card Security

Michael Tunstall

Abstract In this chapter, the various attacks and countermeasures that apply to secure smart card applications are described. This chapter focuses on the attacks that could affect cryptographic algorithms, since the security of many applications is dependent on the security of these algorithms. Nevertheless, how these attacks may be applied to other security mechanisms is also described. The aim of this chapter is to demonstrate that a careful evaluation of embedded software is required to produce a secure smart card application.

Keywords Embedded software · Fault analysis · Side channel analysis · Smart card security

9.1 Introduction

The implementation of secure applications on smart cards is different to the development on other platforms. Smart cards have limited computing power, comparatively small amounts of memory and are reliant on a smart card reader to provide power and a clock. There are security considerations that are specific to smart cards, that need to be taken into account when developing a secure smart card-based application.

In this chapter, attacks that are specific to smart cards, and other devices based around a secure microprocessor, will be described. There are other considerations that need to be taken into account when implementing a secure application, but these are generic and beyond the scope of this chapter.

There are three main types of attack that are considered in smart card security. These are:

1. **Invasive Attacks**: These are attacks that require the microprocessor in a smart card to be removed and directly attacked through a physical means. This class of attacks can, at least in theory, compromise the security of any secure microprocessor. However, these attacks typically require very expensive equipment and

M. Tunstall (✉)
Cryptography Research, 425 Market Street, San Francisco, CA 94105, USA
e-mail: michael.tunstall@cryptography.com

K. Mayes and K. Markantonakis (eds.), *Smart Cards, Tokens, Security and Applications*, DOI 10.1007/978-3-319-50500-8_9

a large investment in time to produce results. Invasive attacks are therefore considered to be primarily in the realm of semiconductor manufacturers and students at well-funded universities.

An example of such an attack would be to place probes on bus lines between blocks of a chip (a hole needs to be made in the chip's passivation layer to allow this). An attacker could then attempt to derive secret information by observing the information that is sent from one block to another.

At its most extreme this type of attack could make use of a focused ion beam to destroy or create tracks on the chips surface. In theory, this could, for example, be used to reconnect fuses. Traditionally, chip manufacturers typically used a test mode where it was possible to read and write to all memory addresses whilst a fuse was present. Once the fuse was blown inside the chip (before the chip left the manufacturer's factory) this mode was no longer available. In modern secure microprocessors this test circuit is typically removed when the chip is cut from the die preventing the attack.

Further information on invasive attacks is available in [3, 32]. More recently, Tarnovsky [52] has made videos on how these attacks are conducted publicly available, and they should be easy to find.

2. **Semi-Invasive Attacks**: These attacks require the surface of the chip to be exposed. An attacker then seeks to compromise the security of the secure microprocessor without directly modifying the chip.

   Some examples of this type of attack include injecting faults using laser or white light [6, 50]. More details on these attacks are given in later sections, and further information can be found in [27].

   The first attempts at analysing the electromagnetic emanations around a microprocessor required a suitable probe to be placed very close to the surface of a targeted microprocessor [17, 45]. This has since been shown to be unnecessary and such analyses can be classed as non-invasive.

3. **Non-Invasive Attacks**: These attacks seek to derive information without modifying a smart card, i.e. both the secure microprocessor and the plastic card remain unaffected. An attacker will attempt to derive information by observing information that leaks during the computation of a given command, or attempts to inject faults using mechanisms other than light.

   Some examples of this type of attack would be to observe the power consumption of a microprocessor [31], or to inject faults by putting a glitch into the power supply [3]. Further descriptions of power analysis attacks can be found in [34], and fault attacks in [27].

This chapter will focus on semi-invasive and non-invasive attacks, as the equipment required to conduct these attacks is more readily available. Invasive attacks are of interest, but are extremely expensive to conduct. This chapter will focus more on what can be achieved in a reasonably funded laboratory. However, some information is given on invasive attacks where relevant.

9.1.1 Organisation

Section 9.2 contains a description of the cryptographic algorithms that will be used in later sections to give examples of attacks. Section 9.3 describes certain hardware security features that are typically included in a smart card. Section 9.4 describes the different forms of side channel analysis and how they can be applied to smart card implementations of cryptographic algorithms. Section 9.5 describes how fault attacks can be applied to smart cards. Section 9.6 describes how the techniques given in Sects. 9.4 and 9.5 can be applied to other security mechanisms. Section 9.7 summarises the chapter.

9.1.2 Notation

The base of a value is determined by a trailing subscript, which is applied to the whole word preceding the subscript. For example, $\texttt{FE}_{16}$ is 254 expressed in base 16 and $d = (d_{\ell-1}, d_{\ell-2}, \ldots, d_0)_2$ gives a binary expression for the ℓ-bit integer d.

In all the algorithms described in this chapter, ϕ represents Euler's totient function, where $\phi(N)$ equals the number of positive integers less than N which are coprime to N. In particular, if $N = pq$ is an RSA modulus then $\phi(N) = (p-1)(q-1)$.

9.2 Cryptographic Algorithms

Some of the attacks detailed in later sections will assume a detailed knowledge of some of the commonly used cryptographic algorithms. The Data Encryption Standard (DES) and RSA, are detailed in this section to provide a reference, and to describe the notation that will be used. For brevity, we do not describe the Advanced Encryption Standard (AES) [43], but mention of how the AES affects security will be made where required.

9.2.1 Data Encryption Standard

The Data Encryption Standard (DES) was introduced by NIST in the mid 1970s [42], and was the first openly available cryptography standard. It became, and still is, a worldwide *de facto* standard for numerous purposes. However, it has been practically demonstrated that an exhaustive search is possible, leading to the introduction of triple DES (see below).

DES can be considered as a transformation of two 32-bit variables (L_0, R_0), i.e. the message block, through sixteen iterations of a round function, as shown in Fig. 9.1, to produce a ciphertext block (L_{16}, R_{16}). The Expansion permutation

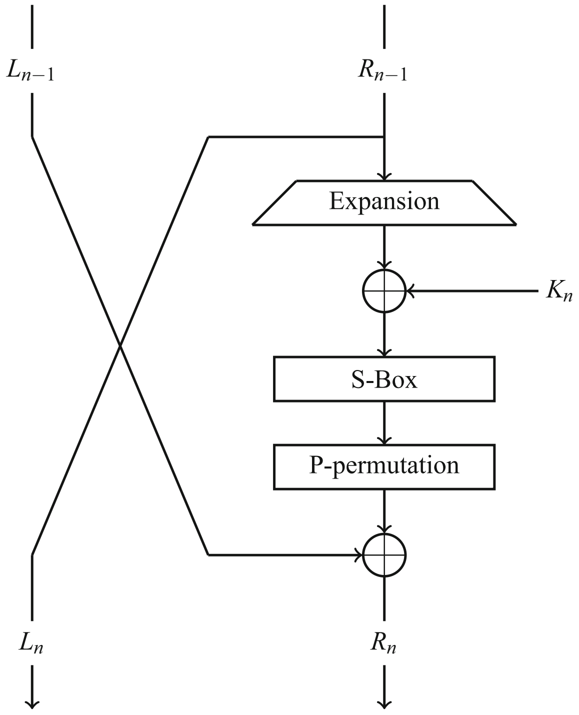


Fig. 9.1 The DES round function for round *n*

selects eight overlapping six-bit substrings from R_n. The P-permutation is a bitwise permutation on the 32-bit output of the S-box function. For clarity of expression, these permutations will not always be considered and the round function will be written as:

$$\begin{aligned} R_n &= S(R_{n-1} \oplus K_n) \oplus L_{n-1} \\ L_n &= R_{n-1} \end{aligned} \tag{9.1}$$

where S is the S-box function. The subkeys K_n, for $1 \leq n \leq 16$, are each 48 bits generated from a 56-bit secret key, by choosing 48 bits from the 56-bit key. This is done by initially conducting a bitwise permutation on the key, referred to as Permuted Choice 1 (PC1). Each round bit shifts are conducted on the key, and 48 bits are chosen from this shifted key to form each subkey using the Permuted Choice 2 (PC2) function.

Eight different S-boxes are applied in each round to sets of six bits, thereby reducing the 48-bit output of the XOR with K_n to 32 bits. Each S-box is a substitution table that is addressed using six bits of information, and each entry is a 4-bit number.

The algorithm also includes an initial and final permutation (these permutations are referred to as IP and IP^{-1} respectively), where the final permutation is the inverse of the initial permutation. More precisely the permutation at the end of DES is conducted on (R_{16}, L_{16}) rather than (L_{16}, R_{16}). These permutations will be ignored in this chapter, as they do not contribute to the security of the algorithm. The permutations IP and IP^{-1} were included because it was the most convenient way of introducing the bits into the first chips used to calculate the DES algorithm (at the time, software implementations were not considered because of the complexity of the algorithm) [38].

9.2.1.1 Triple DES

In order to mitigate the aforementioned key length problem, a modification to DES was proposed to make an exhaustive key search prohibitively complex. Triple DES is a construction that uses two different DES keys and is defined in [42]. In the algorithm below the two DES keys are denoted K_1 and K_2, and in order to generate a ciphertext block C from a plaintext block M the following calculation is performed:

$$C = \mathrm{DES}\,(\mathrm{DES}^{-1}\,(\mathrm{DES}\,(M,\ K_1),\ K_2),\ K_1) \tag{9.2}$$

where $\mathrm{DES}(M, K)$ denotes the output of the DES encryption algorithm applied to message block M with key K. Deciphering the ciphertext block C uses the function,

$$M = \mathrm{DES}^{-1}\,(\mathrm{DES}\,(\mathrm{DES}^{-1}(C,\ K_1),\ K_2),\ K_1) \tag{9.3}$$

The structure of triple DES allows for backward compatibility as if K_1 and K_2 are equal the resulting ciphertext will the equivalent to that produced with a single DES. The triple DES requires that three instantiations of the DES algorithm are used, since it has been shown that two instantiations of DES only increase the security of the algorithm by one bit (see meet-in-the-middle attacks [36]).

Another version of triple DES is also proposed in [42], in which three different keys are used rather than two.

9.2.2 RSA

RSA was first introduced in 1978 [47], and was the first published example of a public key cryptographic algorithm. The security of RSA depends on the difficulty of factorising large numbers, meaning that RSA keys need to be quite large. Advances in factorisation algorithms and the constantly increasing processing power available in modern computers has led to constantly increasing key sizes. At the time of writing RSA is typically used with 1024 or 2048-bit key sizes.

To generate a key pair for use with RSA, two prime numbers, p and q, of a similar bit length, are generated; they are then multiplied together to create a value N, the modulus, whose bit length is equal to that desired for the cryptosystem. That is, in order to create a 1024-bit modulus, $2^{511.5} < p, q < 2^{512}$ [1] (if values of p or q are chosen from the interval $(2^{511}, 2^{511.5})$ the product of p and q is not guaranteed to have a bit length of 1024 bits). A public exponent, e, is chosen that is coprime to both $p - 1$ and $q - 1$.

A private exponent, d, is generated from the parameters previously calculated, using the formula:

$$\begin{aligned} e\,d &\equiv 1 \pmod{(p-1)(q-1)}, \text{ or equivalently} \\ e\,d &\equiv 1 \pmod{\phi(N)} \end{aligned} \tag{9.4}$$

where ϕ is Euler's Totient function.

9.2.2.1 The RSA Cryptosystem

In the RSA cryptosystem, to encrypt a message, M, and create ciphertext, C, one calculates:

$$C = M^e \mod N \tag{9.5}$$

The value of e is often chosen as 3 or $2^{16} + 1$, as these values are small, relative to N, and have a low Hamming weight, which means that the encryption process is fast (see below). To decrypt the ciphertext, the same calculation is carried out but using the private exponent, d, which generally has a similar bit length as N:

$$M = C^d \mod N \tag{9.6}$$

9.2.2.2 The RSA Signature Scheme

The RSA digital signature scheme involves the reverse of the operations used in the RSA cryptosystem. The generation of a signature, S, uses the private exponent d. By convention, this is expressed as:

$$S = M^d \mod N \tag{9.7}$$

The verification therefore uses the public exponent and is expressed as:

$$M = S^e \mod N \tag{9.8}$$

[1]This is possibly overly strong, as it typically recommend that the bit lengths of p and q are approximately equal. However, it will provide the most security for a modulus of a given bit length assuming that $p - q$ is sufficiently large to prevent an attacker from guessing their values by calculating $\sqrt{N}$.

9.2.2.3 Padding Schemes

Applying the RSA primitive to a message, as described above, will not yield a secure signature or encryption scheme (for reasons beyond the scope of this chapter). To achieve a secure scheme it is necessary to apply the RSA operation to a transformed version of the message, e.g. as can be achieved by hashing the message, adding padding, and/or masking the result. This process is termed padding, and the interested reader is referred to [36] for a treatment of padding schemes.

Some of the attacks presented in this chapter will not be realistic when a padding scheme is used, since padding schemes mean that an attacker cannot entirely control a message. However, it is important that an implementation of RSA is secure against all possible attacks. If a given implementation does not use padding or, more realistically, contains a bug that allows an attacker to remove the padding function, the implementation should still be able to resist all the attacks described in this chapter.

9.2.2.4 Computing a Modular Exponentiation

Many different algorithms can be used to calculate the modular exponentiation algorithm required for RSA. In practice, a large number of algorithms cannot be implemented on smart cards, as the amount of available memory does not usually allow numerous intermediate values to be stored in RAM. The manipulation of large numbers is typically performed using a coprocessor (see Sect. 9.3), as implementing a multiplication on an 8-bit platform would not give a desirable performance level.

The simplest exponentiation algorithm is the square and multiply algorithm [36], and is given in Algorithm 1 for an exponent d of bit length ℓ:

Algorithm 1: The Square and Multiply Algorithm

```
Input: $M, d = (d_{\ell-1}, d_{\ell-2}, \ldots, d_0)_2, N$
Output: $S = M^d \bmod N$

$A \leftarrow 1$ ;
for $i = \ell - 1$ to $0$ do
    $A \leftarrow A^2 \bmod N$ ;
    if $(d_i = 1)$ then
        $A := A \cdot M \bmod N$ ;
    end
end
return $A$
```

The square and multiply algorithm calculates $M^d \bmod N$ by loading the value one into the accumulator A and d is read bit-by-bit. For each bit a squaring operation modulo N takes place on A, and when a bit is equal to one A is subsequently multiplied by M. It is because of this multiplication that e is typically chosen as 3 or $2^{16} + 1$, as both values only have two bits set to one; therefore minimising the number of multiplications required. It is not possible to only have one bit set to one as it is necessary for e to be an odd number in order for it to have an inverse modulo N. The most significant bit of a number will always be set to one, and the least significant bit will need to be set to one to produce an odd number.

9.2.2.5 Using the Chinese Remainder Theorem

The RSA calculation using the private exponent (i.e. where $S = M^d \bmod N$ and $N = p \cdot q$) can be performed using the Chinese Remainder Theorem (CRT) [29]. Initially, the following values are calculated,

$$\begin{aligned} S_p &= M^{d \bmod (p-1)} \bmod p \\ S_q &= M^{d \bmod (q-1)} \bmod q \end{aligned} \tag{9.9}$$

which can be combined to form the RSA signature S using the formula $S = a\, S_p + b\, S_q \bmod N$, where:

$$\begin{aligned} a &\equiv 1 \pmod p \\ a &\equiv 0 \pmod q \end{aligned} \quad \text{and} \quad \begin{aligned} b &\equiv 0 \pmod p \\ b &\equiv 1 \pmod q \end{aligned}$$

This can be implemented in the following manner:

$$S = S_q + \left(\left(S_p - S_q\right) q^{-1} \bmod p\right) \cdot q \tag{9.10}$$

This provides a method of calculating an RSA signature that is approximately four times quicker than a generic modular exponentiation algorithm, i.e. two exponentiations, each of which can be computed eight times faster than an exponentiation using d (the bit length of $d \bmod (p-1)$ and $d \bmod (p-1)$ will be half that of d). This advantage is offset by an increase in the key information that needs to be stored. Rather than storing just the value of d and N, the values of $(p, q, d \bmod (p-1), d \bmod (q-1), q^{-1} \bmod p)$ need to be precalculated and stored.

9.3 Smart Card Security Features

This section will detail some of the features of smart cards that are pertinent when considering their security. Smart cards have traditionally been based on 8-bit Complex Instruction Set Computer (CISC) architectures [40]. Usually built around a Motorola 6805 or Intel 8051 core, often with extensions to the instruction set. More sophisticated smart cards are emerging based on 32-bit Reduced Instruction Set Computer (RISC) architecture chips, containing dedicated peripherals (cryptographic coprocessors, memory managers, large memories, ...) [39].

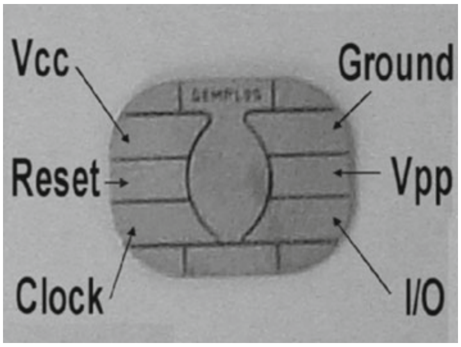


Fig. 9.2 The Contacts used to power and communicate with a smart card

9.3.1 *Communication*

A smart card has five contacts that it uses to communicate with the outside world defined in the ISO/IEC 7816-2 standard [25]. Two of these are taken by the power supply (usually 3 or 5 v), referred to as Vcc, and the ground used to power the chip. Another contact is used to supply a clock, which is allowed to vary between 1 and 5 MHz but is typically set to 3.57 MHz. The remaining two contacts are used to communicate with the microprocessor. A sixth contact was originally used to provide a higher voltage to program the EEPROM (referred to as Vpp), but is no longer in use for reasons described in Sect. 9.6. The location of the different contacts is shown in Fig. 9.2.

One of the contacts, referred to as the I/O, is used for communication and to send commands to the chip in a smart card. The protocols used to communicate with a smart card are referred to as T = 0 and T = 1 and are defined in the ISO/IEC 7816-3 standard [24]. This section will describe both protocols, as they are nearly identical. The extra requirements of T = 1 are detailed where relevant.

The remaining contact is used to reset the smart card (there are a further 2 contacts defined in the ISO/IEC 7816-3 standard but they are not currently used). Resetting a smart card is a physical event (i.e. moving the voltage applied to this contact from 0 to 1) and it will always provoke a response from a smart card. A user can apply the reset at any time. The smart card will respond by sending an Answer To Reset (ATR) via the I/O contact, which is a string of bytes that defines the protocols the smart card can use, the speeds at which the smart card can communicate and the order in which bits are going to be sent during the session (i.e. most or least significant bit first).

9.3.2 *Cryptographic Coprocessors*

Traditionally, smart cards have been based around 8-bit architectures. In order to manipulate large numbers, e.g. to calculate the RSA algorithm, dedicated coprocessors can be appended to the CPU. In more modern 32-bit chips [39] this is not always necessary, as efficient software implementations can be achieved. However, coprocessors are still often used for increased performance. DES and AES is also often implemented in a coprocessor to help increase performance, where hardware implementations of these secret key algorithms can typically be expected to require one or two clock cycles per round of the block cipher. The inclusion of coprocessors increases the size of a microprocessor and the overall power consumption. This means that chips with coprocessors are usually more expensive and are not always ideal in environments where the amount of available current is limited.

9.3.3 Random Number Generators

Random number generators are usually included in smart cards, as unpredictable numbers are an important element in many secure protocols. A "true" random number generator is typically based on a signal generated by an analogue device which is then treated to remove any bias that may exist, or has been induced, in the bits generated. The correct functioning of all aspects of a smart card chip under varied environmental conditions is important, but is critical for random number generation because the quality of the generated random values can have a profound effect on cryptographic schemes. Random number generators are therefore designed to function correctly in a large range of environmental conditions, including temperature, supply voltage, and so on. However, if an attacker succeeds in modifying the environmental conditions such that the physical source of randomness is affected, the output is typically treated such that an attacker will not be able to determine if the change in conditions had any effect.

Pseudo-random number generators are also often included in a secure microprocessor. These are typically based on Linear Feedback Shift Registers (LFSRs) that are able to generate a new pseudo-random value every clock cycle, but are deterministic over time and are not usually used for critical security functions.

Where random values are required in cryptographic algorithms, a true random number generator is used when the quality of the random value is important, e.g. for use in a cryptographic protocol. Where the quality of the random value is less important, a pseudo-random number generator can be used. In some secure microprocessors only pseudo-random number generators are available. In this case, mechanisms that combine a random seed (that can be inserted into the chip during manufacture) with pseudo-random values can be used to provide random values.

An example of this latter type of random number generator is given in the ANSI X9.17 [2, 22] standard, that uses DES to provide random values based on a random seed generated during the initialisation of a given device and another source of pseudo-random information. This functions by taking a 64-bit pseudo-random input (X), a 64-bit random seed (S) and a DES key (K). X is usually generated by calculating $X = \text{DES}(D, K)$, where D is a the date and/or time, but this information is not available to a smart card and is typically replaced with values provided by a pseudo-random number generator. To output a random value R the following calculation takes place:

$$R = \mathrm{DES}(X \oplus S, K), \tag{9.11}$$

and the random seed is updated using:

$$S = \mathrm{DES}(R \oplus X, K). \tag{9.12}$$

For increased security the DES function can be replaced with triple DES, as the key length used by DES has proven to be too short to entirely resist an exhaustive key search.

9.3.4 Anomaly Sensors

There are usually a number of different types of anomaly detectors present in smart cards. These are used to detect unusual events in the voltage and clock supplied to the card, and the environmental conditions (e.g. the temperature). These enable a smart card to detect when it is exposed to conditions that are outside the parameters within which it is known to function correctly. When unusual conditions are detected, the chip will cease to function until the effect has been removed (i.e. initiate a reset or execute an infinite loop when the sensor is activated). However, it is considered prudent not to rely solely on these sensors and to implement further countermeasures (see Sect. 9.5).

9.3.5 Chip Features

The surface of the chip used in a smart card can be uncovered by removing the plastic body of the card and using fuming nitric acid to remove the resin used to protect the microprocessor. Once the chip has been revealed the easiest form of analysis is to simply look at it under a microscope. The various different blocks can often be identified, as shown in Fig. 9.3.

Reverse engineering can target the internal design to understand how a given chip or block functions. An attacker can use such information to improve their knowledge of chip design and find potential weaknesses in the chip, which may allow them to compromise the chip's integrity.

In modern smart cards, various features used to inhibit reverse engineering are implemented using glue logic: important blocks are laid out in a randomised fashion that makes reverse engineering difficult. This technique increases the size of the block, and is therefore not used in the design of large blocks such as Read-Only Memory (ROM) and Electrically Erasable Programmable Read-Only Memory (EEPROM).

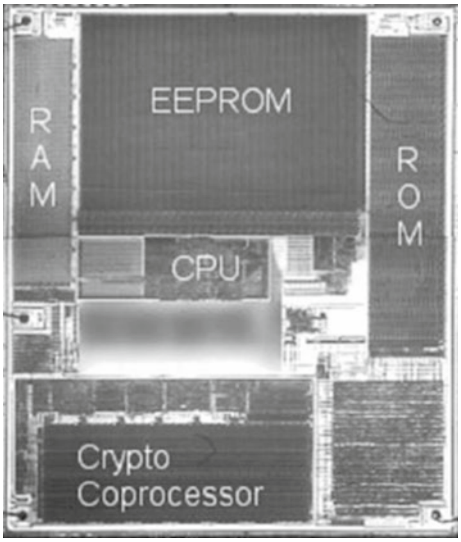


Fig. 9.3 A chip surface with readily identifiable features

Another common technique to prevent this sort of identification and targeting is to overlay the chip with another metal layer that prevents the chip's features being identified. This can be removed using hydrofluoric acid that eats through the metal layer; this reaction is then stopped using acetone before further damage is done and the chip surface can be analysed. The chip becomes non-functional but the layout of the chip can be determined, so that other chips of the same family can be attacked. The result of such a process is shown in Fig. 9.4.

Discovering the layout and functioning of a chip is particularly important when using a laser as a fault injection mechanism (see Sect. 9.5). Different areas of a chip can be targeted through the metal layer once the layout of a chip is known. Tarnovsky [52] has made videos on how these attacks are conducted publicly available and they should be easy to find.

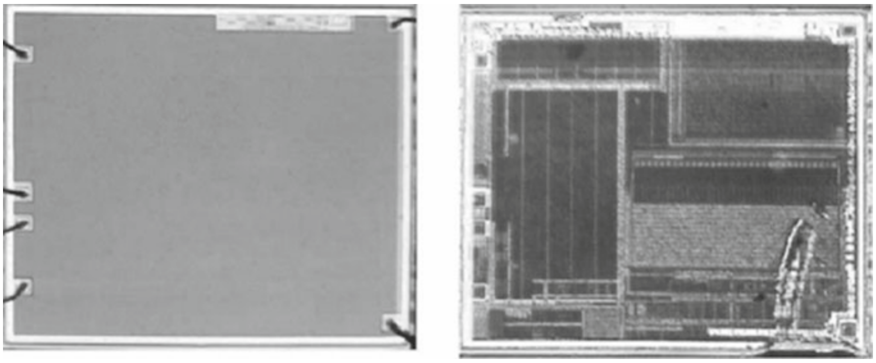


Fig. 9.4 A chip with a shield present and removed

9.4 Side Channel Analysis

Side-channel attacks are a class of attacks, where an attacker will attempt to deduce what is occurring inside a device by observing information that leaks during the normal functioning of the device. If this information can be related to any secret information being manipulated the security of the device can be compromised. It should be noted that side channel analysis is a passive form of attack, i.e. an attacker will simply observe what is occurring when a device is functioning. In the case of smart cards the message being manipulated can be controlled, but this is not necessary to construct a valid side channel attack.

The first publication that mentions a side-channel attack is described in [53]. In 1956, MI5 mounted an operation to decipher communications between Egyptian embassies. The communications were enciphered using Hagelin machines [28]. These machines did not function using a key value as described in Sect. 9.2. Enciphering occurred by routing electronic signals from a keyboard through seven rotating wheels to generate a ciphertext. The "key" was the initial setting of these seven wheels. The machine was reset every morning by the clerk who would be sending messages. MI5 managed to plant a microphone in close proximity to one of these machines. This allowed the initial settings to be determined by listening to the initial settings being made every morning. This would have allowed them to decipher intercepted communications with another Hagelin machine set to the same key. In practice, MI5 was only able to determine a certain amount of wheel settings because of the difficulty of distinguishing the noise of the wheels being set from background noise. This made deciphering more complex, but not impossible, as the number of possible keys was significantly reduced by the partial information.

9.4.1 Timing Analysis

The first modern example of a side channel attack was proposed in [30]. This involved observing the differences in the amount of time required to calculate an RSA signature for different messages to derive the secret key. This attack was conducted against a PC implementation, but a similar analysis could potentially be applied to smart card implementations. It would be expected to be more efficient against a smart card as more precise timings can be achieved with an oscilloscope or proprietary readers. An example of a trace acquired with an oscilloscope that would provide this sort of information is shown in Fig. 9.5. The I/O events on the left-hand side of the figure represent the reader sending a command to the smart card. The I/O events on the right hand side of the figure show the response of the smart card. The time taken by a given command can be determined by observing the amount of time that passes between these two sets of events.

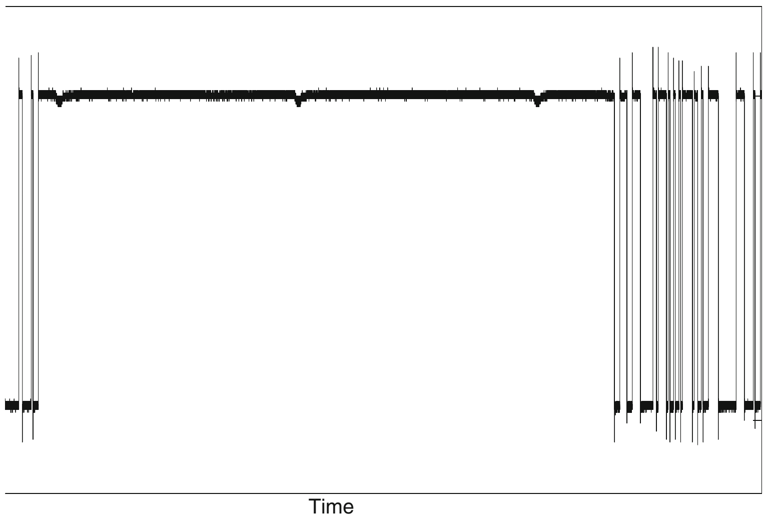


Fig. 9.5 The I/O of a smart card command

9.4.2 Power Analysis

The most common form of side-channel attack, when considering smart cards, is the analysis of the instantaneous power consumption [31]. This is measured by placing a resistor in series with a smart card and the power supply (or ground), and measuring the potential difference across the resistor with an oscilloscope. The acquired information can be analysed *a posteriori* to attempt to determine information on what is occurring within a secure microprocessor. There are two main types of power attack; these are Simple Power Analysis (SPA) and Differential Power Analysis (DPA).

9.4.2.1 Simple Power Analysis

A powerful form of power analysis is to search for patterns within an acquired power consumption trace. An attacker can attempt to determine the location of individual functions within a command. For example, Fig. 9.6 shows the power consumption of a smart card during the execution of DES. A pattern can be seen that repeats 16 times, corresponding to the 16 rounds that are required during the computation of DES.

This analysis can be further extended by closely inspecting the power consumption during the computation of one round, to attempt to determine the individual functions within each round. This is shown in Fig. 9.7, where the functions in the second round of a DES implementation are evident in the power consumption trace. This may not

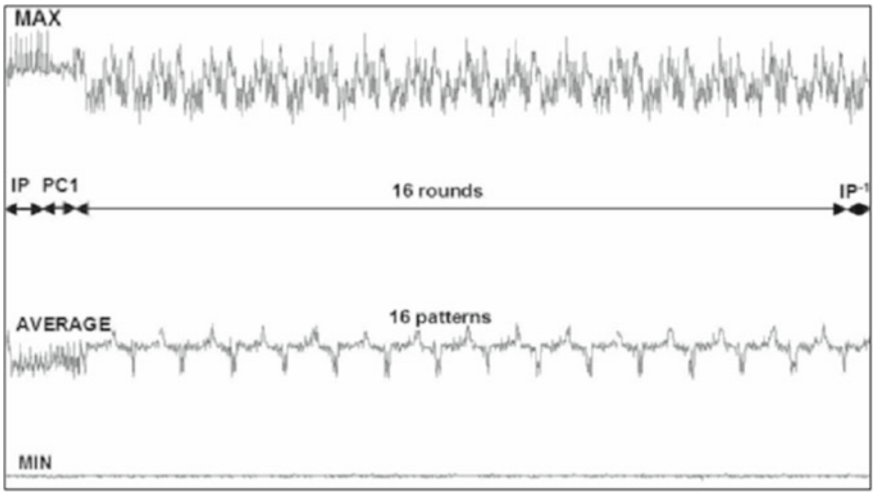
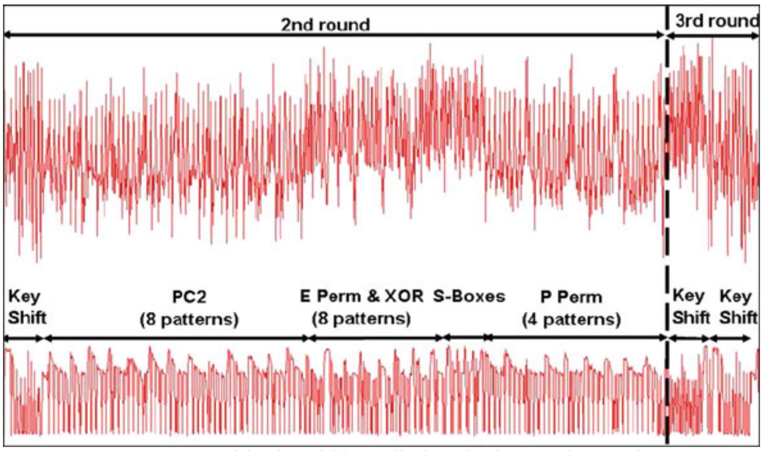


Fig. 9.6 The power consumption of a DES implementation showing the rounds of the algorithm



Key:	**Key Shift**	A bitwise shift applied to the key each round.
	PC2	PC2 used to generate a 48-bit round key each round.
	E Perm	Expansion permutation applied to R_i, for $1 \leq i \leq 16$, to produce a 48-bit output.
	XOR	The XOR with the round key.
	S-boxes	Eight substitution tables reducing 48 bits to 32 bits.
	P Perm	The P permutation, a bitwise transformation.

Fig. 9.7 The power consumption of a DES implementation showing the round functions

be immediately apparent, as close inspection of the trace's features is necessary to identify the individual functions. For example, if an attacker is seeking to determine where the compression permutation (PC2) is computed, they will look for eight patterns of six events. This is because the compression permutation selects 48 bits from the 56-bit DES key, where the 48-bit result is divided into segments of 6 bits (for use in the S-box function). The natural method of implementing this permutation will, therefore, be to construct a loop that will repeat eight times. Each loop will move 6 bits from the DES key to the 48-bit output. This should therefore produce eight patterns of six events because of the individual bits being selected and written.

The use of this information is not necessarily immediately apparent; an attacker can use this information to improve the effectiveness of other attacks. The efficiency of the statistical treatment required for Differential Power Analysis (DPA) [31] can be increased by taking more precise acquisitions. This is because the area that needs to be analysed can be defined, and therefore reducing the amount of data that needs to be acquired. A more detailed analysis is given below.

The same is true for fault injection techniques, detailed in Sect. 9.5, as it is often necessary to target specific events. If arbitrary functions can be identified using the power consumption, the point in time at which an attacker wishes to inject a fault can be discovered. This can greatly decrease the time required to conduct a successful attack, as less time is wasted injecting faults into areas of the computation that will not produce the desired result.

The examination of the power consumption can also be used to determine information on the private/secret keys used in a naïve implementations of cryptographic algorithms. For example, if the power consumption of a smart card during the generation of an RSA signature using the square and multiply algorithm is analysed, it may be possible to determine some bits of the private key. An example of the power consumption during the generation of an RSA signature is shown in Fig. 9.8.

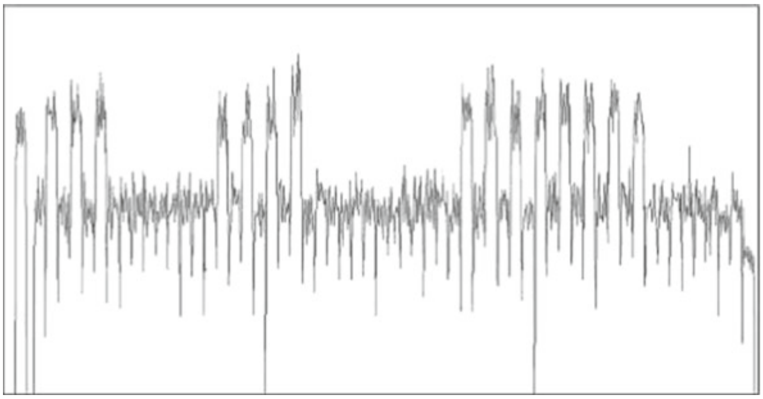


Fig. 9.8 The power consumption of an RSA implemented using the square and multiply algorithm

Looking closely at the acquired power consumption, a series of events can be seen. There are two types of events at two different power consumption levels, with a short dip in the power consumption between each event. This corresponds well to the square and multiply algorithm described in Sect. 9.2. Given the ratio of the two features, it can be assumed that the feature with the lower power consumption represents the squaring operation and the higher power consumption represents the multiplication. From this, the beginning of the exponent can be read from the power consumption, in this case the exponent used is $\texttt{F00F000FF00}_{16}$.

It should be noted that all the examples given in this section have been taken from chips that display the differences in an obvious manner. Modern secure microprocessors rarely display the functions being executed as clearly as in the examples given.

9.4.2.2 Differential Power Analysis

The idea of statistically treating power analysis traces was first presented to the cryptographic community in [31], and is referred to as Differential Power Analysis (DPA). DPA is based on the relationship between the power consumption and the data being manipulated at a given point in time. The differences in power consumption are potentially extremely small, and cannot be interpreted individually, as the information will be lost in the noise incurred during the acquisition process. The small differences produced can be seen in Fig. 9.9, where traces were taken using a chip where the acquisition noise is exceptionally low. Different power levels, corresponding to different Hamming weights of the data being manipulated, are clearly visible.

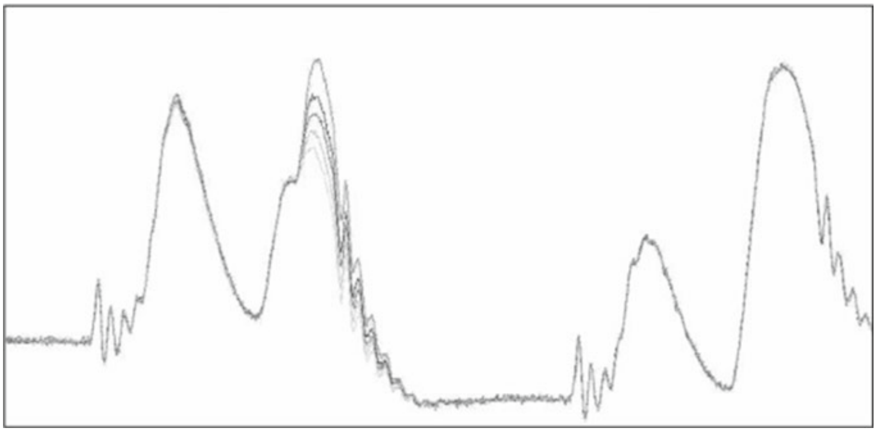


Fig. 9.9 Overlaid acquisitions of the power consumption produced by the same instruction but with varying data

Differential Power Analysis (DPA) can be performed on any algorithm in which an intermediate operation of the form $\beta = S(\alpha \oplus K)$ is calculated, where α is known and K is the key (or some segment of the key). The function S is typically a nonlinear function, usually a substitution table (referred to as an S-box), which produces an intermediate output value β.

The process of performing the attack initially involves running a microprocessor N times with N distinct message values M_i, where $1 \leq i \leq N$. The encryption of the message M_i under the key K to produce the corresponding ciphertext C_i will result in power consumption traces w_i, for $1 \leq i \leq N$. These traces can be captured with an oscilloscope, and sent to a computer for analysis and processing.

To find K, one bit of β is chosen, which we will refer to as b. For a given hypothesis for K, this bit will classify whether each trace w_i is a member of one of two possible sets. The first set S_0 will contain all the traces where b is equal to zero, and the second set S_1 will contain all the remaining traces, i.e. where the output bit b is equal to one.

A differential trace Δ_n is calculated by finding the average of each set and then subtracting the resulting values from each other, where all operations on traces are conducted in a pointwise fashion, i.e. this calculation is conducted on the first point of each acquisition to produce the first point of the differential trace, the second point of each acquisition to produce the second point of the differential trace, etc.

$$\Delta_n = \frac{\sum_{w_i \in S_0} w_i}{|S_0|} - \frac{\sum_{w_i \in S_1} w_i}{|S_1|}$$

A differential trace is produced for each value that K can take. In DES the first subkey will be treated in groups of six bits, so 64 (i.e. 2^6) differential traces will be generated to test all the combinations of six bits. The differential trace with the highest peak will validate a hypothesis for K, i.e. $K = n$ corresponds to the Δ_n featuring a maximum amplitude. An example of a differential trace produced by predicting one bit of the output a DES S-box, with a correct key guess, is shown in Fig. 9.10.

The differential trace in Fig. 9.10 shows a large difference in the power consumption at five different points, which are referred to as DPA peaks. The first peak

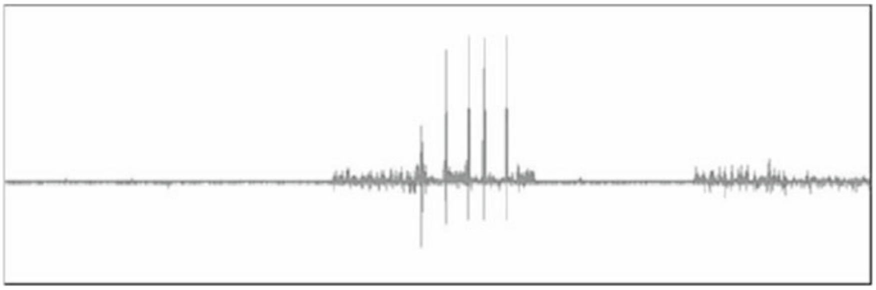


Fig. 9.10 A differential trace

corresponds to the output of the S-box, i.e. where the output of the S-box function is determined and written to memory. The four subsequent peaks correspond to the same bit being manipulated in the P-permutation. This occurs because the output of each S-box consists of four bits, and the memory containing those four bits will be accessed each time one of those bits is required in the output of the P-permutation.

A more complete version of this attack uses Pearson's correlation coefficient to demonstrate the correlation between the Hamming weight and the instantaneous power consumption. This can be used to validate key hypotheses in an identical manner to DPA. Details of this method are beyond the scope of this chapter, but the interested reader is referred to [11].

9.4.3 Electromagnetic Analysis

An alternative side channel to measuring the power consumption of a smart card is to measure the instantaneous electromagnetic emanations as a cryptographic algorithm is being computed [17, 45]. This is typically implemented using a small probe, an example of which can be seen in Fig. 9.11. Such probes can measure the electromagnetic emanations for different blocks of a chip, as such probes are an equivalent size to the chip's features. This means that the probe can be placed just above a given feature to try and get a strong signal from that part of the chip, e.g. the bus between two areas of the chip, while excluding noise from other areas of the chip.

Measuring the electromagnetic field can be done using a handmade probe (such as that shown in Fig. 9.11), although commercially available probes are also sufficient (and has the advantage that the frequency response is defined). The signals from a probe is passed through an amplifier and can be acquired using an oscilloscope in the same way one would acquire a power consumption trace.

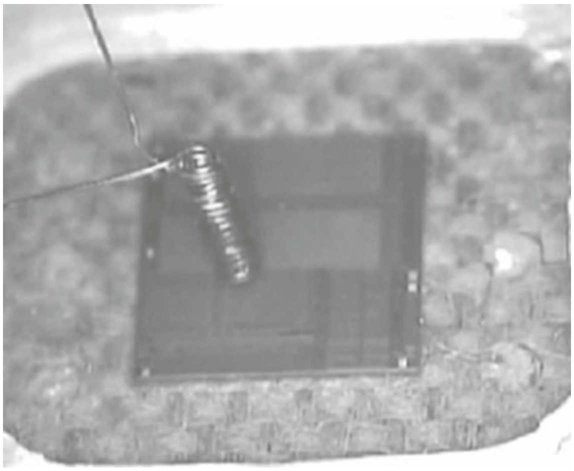
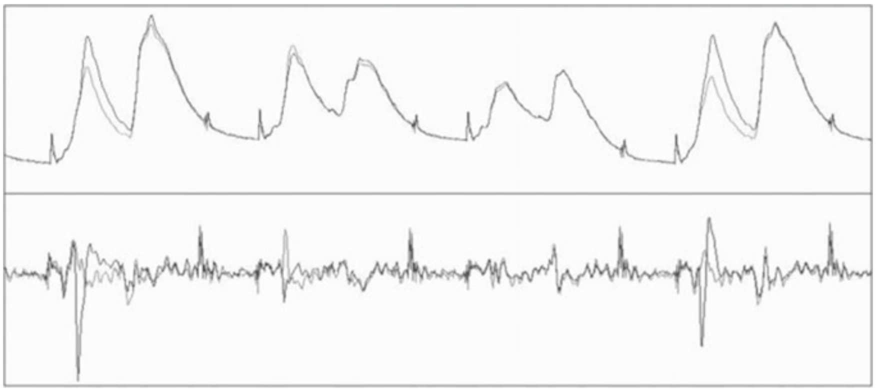


Fig. 9.11 Electromagnetic probing of a chip



Key: The upper traces represents the power consumption, and the lower traces represent the electromagnetic emanations during the same command. The black traces were taken were $\texttt{FF}_{16}$ is being manipulated, and the grey traces where $\texttt{00}_{16}$ is being manipulated.

Fig. 9.12 Power and electromagnetic measurements

The initial descriptions of attacks were based on acquired traces of the electromagnetic emanations of a microprocessor, as shown in Fig. 9.11. More recent descriptions have shown that at attacker can find low frequency leakage in complex System-on-Chip microprocessors [33] and even the metal case of a Hardware Security Module (HSM) or PC [18]. In these cases, an attack was realised by extracting a signal at a well-chosen frequency by filtering the signal before acquisition with an oscilloscope.

The signals that are acquired using this method are also different to those acquired by reading the instantaneous power consumption. The signals acquired during two executions of a selected command by an 8-bit microprocessor is shown in Fig. 9.12. The black traces show the acquired power and electromagnetic signals when the chip manipulates $\texttt{FF}_{16}$, and the grey traces shows the same command where the microprocessor is manipulating $\texttt{00}_{16}$. The difference in the black and grey traces representing the power consumption can be seen as a increase in the power consumption for short periods. The difference in the traces representing the electromagnetic emanations is caused by sudden changes in the electromagnetic field, shown by spikes in the signal at the same moment in time the difference in the power consumption can be observed.

The traces acquired from measuring the instantaneous electromagnetic emanations can be treated in exactly the same way as power consumption acquisitions [17, 45]. The acquisitions can be analysed individually, referred to as Simple ElectroMagnetic Analysis (SEMA), or treated statistically, referred to as Differential ElectroMagnetic Analysis (DEMA).

9.4.3.1 Leakage Detection

Differential power analysis, and similar attacks, target specific intermediate values generated during the computation of a cryptographic algorithm. When developing a side channel resistant implementation of a cryptographic implementation one needs to be able to determine that no attack is possible from *any* intermediate state. Initial methods to determine if an implementation could be attacked via side-channel analysis would attempt to attack the commonly targeted intermediate states.

Recent results have suggested that an implementation of a cryptographic algorithm can be tested, by determining if the leakage acquired during the processing of a fixed input can be distinguished from the leakage acquired during the processing of a random input [20].

One of the tests in the Test Vector Leakage Assessment (TVLA) methodology is to determine whether there are statistically significant differences in the mean traces of two sets of traces, one acquired with a fixed plaintext and the other with random plaintexts. In applying this, one would take two sets of data, and conduct Welch's t-test point-by-point to determine whether there is evidence against the null hypothesis that the sets are the same.

Consider two sets of acquisitions, of n_1 and n_2 samples, respectively. We can compute their respective sample means, $\bar{X}_1$ and$\bar{X}_2$, and respective sample standard deviations, σ_1 and σ_2. One can then compute a t-statistic using Welch's t-test:

$$\alpha = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{{\sigma_1}^2}{n_1} + \frac{{\sigma_2}^2}{n_2}}} \;, \tag{9.13}$$

where the result is distributed over a t-distribution with ν degrees of freedom, i.e. $\alpha \sim t(\nu)$. In practice, one would use the asymptotic result where the t-distribution is equivalent to the standard normal distribution, so ν does not need to be defined.

Goodwill et al. use $\alpha > 4.5$ to indicate the presence of leakage. Specifically, an $\alpha > 4.5$ gives the probability of indicating leakage where no leakage is present, often referred to as a Type I error, of $\sim 1 \times 10^{-6}$.

Further results on how this type of test can be used are given in [49].

9.4.4 Countermeasures

There are several countermeasures for protecting cryptographic algorithms against side-channel attacks. Some countermeasures can either be implemented in hardware or software; only software implementations are considered here for simplicity. These countermeasures are listed below:

Constant Execution can be used to fix the time taken by an algorithm, so that no deductions on secret information can be made though timing analysis or SPA. This extends to individual processes being executed by a smart card. If a process takes different lengths of time depending on some secret information and the difference in time is made up by a dummy function, there is a good chance that this will be visible in the power consumption or electromagnetic emanations. It is, therefore, important that an algorithm is written so that the same code is executed for all the possible input values.

Random Delays can be inserted at different points in the algorithm being executed, i.e. a dummy function that takes a random amount of time to execute can be called. The algorithm can no longer be said to comply with the constant execution criteria given above, but any variation is completely independent of any secret information. This does not prevent any attacks, but creates an extra step for an attacker. In order to conduct any power analysis, an attacker needs to synchronise the power consumption acquisitions *a posteriori*. The problem of attempting to conduct side-channel attacks in the presence of random delays is described in [14].

Randomisation (or data whitening) is where the data is manipulated in such a way that the value present in memory is always masked with a random value. This randomisation remains constant for one execution, but will vary from one acquisition to another. This mask is then removed at the end of the algorithm to produce the ciphertext. Some ideas for implementing this countermeasure were proposed in [12], and an example of this sort of implementation applied to block ciphers can be found in [1].

The size of the random value used in block ciphers is generally limited as S-boxes need to be randomised before the execution of the block cipher. This is generally achieved by creating an alternative S-box in memory for each execution of the cryptographic algorithm using the algorithm given in Algorithm 2.

Algorithm 2: Randomising S-box Values

```
Input: $S = (s_0, s_1, s_2, \ldots, s_n)_x$ containing the S-box, $\mathbf{R}$ a random $\in \{0, 1, \ldots, n\}$, and $r$ a
       random $\in \{0, 1, \ldots, x-1\}$
Output: $RS = (rs_0, rs_1, rs_2, \ldots, rs_n)_x$

for $i = 0$ to $n$ do
    $rs_i \leftarrow s_{(i \oplus \mathbf{R})} \oplus r$ ;
end
return $RS$ ;
```

The random value used for masking the input data can be no larger than n, and the random value used for the output value can be no larger that x. In an implementation of DES $\mathbf{R} \in \{0, 1, \ldots, 63\}$ and $r \in \{0, 1, \ldots, 15\}$, the rest of the algorithm needs to be a carefully designed to produce values masked with R, and to be able to manipulate returned values masked with r.

This is not possible in the case of RSA, where the calculation methods do not facilitate the method described above. A method for randomising the calculation of an RSA signature is given in [26], where the signature generation can be calculated using the formula:

$$S = \left((M + r_1 \cdot N)^{d+r_2 \cdot \phi(N)} \bmod (r_3 \cdot N)\right) \bmod N \tag{9.14}$$

where ϕ is Euler's totient function and, r_1, r_2 and r_3 are small random values. The effect of the each of the small random values does not change the outcome, but the order of the squaring operations and multiplications required to compute S is randomised. This does not provide a totally secure algorithm as the modular exponentiation itself also has to be secured against SPA attacks. A discussion of these algorithms is given in [13].

Randomised Execution is the manipulation of data in a random order so that an attacker does not know what is being manipulated at a given moment in time. If, for example, n bytes are being XORed with n key bytes then it is prudent to do it in a random order. If an attacker wishes to determine which byte has been XORed at any particular time this will be infeasible given that the order that bytes are being manipulated is unknown.

This also inhibits any statistical analysis of a side channel (i.e. using DPA), as this relies on the same unknown variable being treated at the same point in time. As an attacker cannot know the order in which the data has been treated, this provides an extremely efficient countermeasure when combined with randomisation. A discussion of this technique applied to DES is described in [37].

Limiting Key Usage can prevent an attack, since conducting a statistical attack, such as Differential Power Analysis, requires a certain number of traces with a fixed key. An effective countermeasure is to limit the number of uses of a given key to a threshold lower than that required to conduct a side channel attack. However, using this countermeasure places a burden on the smart card as keys need to be updated and some mechanism for synchronising the key with a server needs to be implemented. We refer the reader to [15] for further discussion of this countermeasure.

9.4.4.1 Remarks

The above list gives the countermeasures that would need to be applied to a cryptographic algorithm to render it secure against side-channel analysis. An attacker would, therefore, have to overcome the combination of all these countermeasures. For an extensive treatment of side-channel analysis, the interested reader is referred to [34].

9.5 Fault Analysis

The problem of faults occurring in microprocessors has existed for a relatively long time. One of the initial observations of faults being provoked in microprocessors was accidental. It was observed that radioactive particles produced by elements naturally present in packaging material caused faults in chips [35]. Specifically, these faults were caused by Uranium-235, Uranium-238 and Thorium-230 residues present in the packaging decaying to Lead-206 and releasing α particles. These particles were energetic enough to cause bits stored in RAM to change.

Further research involved the analysis of the effect of cosmic rays on semiconductors [54]. While cosmic rays are very weak at ground level, their effect in the upper atmosphere and outer space is important for the aero-spacial industry. This provoked research into integrity measures that need to be included in semiconductors used in the upper atmosphere and space.

In 1997, it was pointed out that a fault present in the generation of an RSA signature, computed using the Chinese Remainder Theorem, could reveal information on the private key [10] (this attack is detailed below). This led to further research into the effect of faults on the security of implementations of cryptographic algorithms in secure microprocessors, and the possible mechanisms that could be used to inject faults in a microprocessor.

9.5.1 Fault Injection Mechanisms

There are a variety of different mechanisms that can be used to inject faults in microprocessors. These are listed here:

Variations in Supply Voltage [3, 9] during execution may cause a processor to misinterpret or skip instructions.

Variations in the External Clock [3, 4, 32] may cause data to be misread (the circuit tries to read a value from the data bus before the memory has time to latch out the correct value) or an instruction miss (the circuit starts executing instruction $n + 1$ before the microprocessor has finished executing instruction n).

Extremes of Temperature [10, 21] may cause unpredictable effects in microprocessors. When conducting temperature attacks on smart cards, two effects can be obtained [6]: the random modification of RAM cells due to heating, and the exploitation of the fact that read and write temperature thresholds do not coincide in most Non-Volatile Memories (NVMs). By tuning the chip's temperature to a value where write operations work but read operations do not, or the other way around, a number of attacks can be mounted.

Laser Light [16, 23, 44] can be used to simulate the effect of cosmic rays in microprocessors. Laser light is used to test semiconductors that are destined to be used in the upper atmosphere or space. The effect produced in semiconductors is based on the photoelectric effect, where light arriving on a metal surface will induce a current. If the light is intense, as in laser light, this may be enough to induce a fault in a circuit.

White Light [3] has been proposed as an alternative to laser light to induce faults in microprocessors. This can be used as a relatively inexpensive means of fault induction [50]. However, white light is not directional and cannot easily be used to illuminate small portions of a microprocessor.

Electromagnetic flux [48] has also been shown to be able to change values in RAM, as eddy currents can be made strong enough to affect microprocessors. However, this effect has only been observed in insecure microprocessors.

9.5.2 Modelling the Effect of a Fault

The fault injection methods described above may have many different effects on silicon. They can be modelled in ways that depend on the type of fault injection that has been used. The following list indicates the possible effects that can be created by these methods:

Resetting Data: an attacker could force the data to the blank state, i.e. reset a given byte, or bytes, of data back to `00` or $\texttt{FF}_{16}$, depending on the logical representation of, for example, RAM cells.

Data Randomisation: an attacker could change the data to a random value. However, the adversary does not control the random value, and the new value of the data is unknown to the adversary.

Modifying Opcodes: an attacker could change the instructions executed by the chip's CPU, as described in [3]. This will often have the same effect as the previous two types of attack. Additional effects could include removal of functions or the breaking of loops. The previous two models are algorithm dependent, whereas the changing of opcodes is implementation dependent.

These three types of attack cover everything that an attacker could hope to do to an implementation of an algorithm. It is not usually possible for an attacker to create all of these possible faults in any particular implementation. Nevertheless, it is important that algorithms are able to tolerate all types of fault, as the fault injection methods that may be realisable on a given platform are unpredictable. While an attacker might only ever have a subset of the above effects available, if that effect is not taken into account then it may have catastrophic consequences for the security of a given implementation.

In the literature one-bit faults are often considered. This is a useful model for developing theoretical attacks, but has proven to be extremely difficult to produce on a secure microprocessor. The model given above is based on published descriptions of implementations of fault attacks.

9.5.3 Faults in Cryptographic Algorithms

The faults mechanisms and fault model described above can be used to attack numerous cryptographic algorithms. Two examples of fault attacks on cryptographic algorithms are described below.

9.5.3.1 Faults in RSA Signature Generation

The first published fault attack [10], proposed an attack focused on an implementation of RSA using the Chinese Remainder Theorem (CRT). The attack allows for a wide range of fault injection methods, as it only requires one fault to be inserted in order to factorise the RSA modulus.

The technique requires an attacker to obtain two signatures for the same message, where one signature is correct and the other is the result of the injection of a fault during the computation of S_p or S_q (see above). That is, the attack requires that one of S_p and S_q is computed correctly, and the other is computed incorrectly.

Without loss of generality, suppose that $S' = aS_p + bS'_q \bmod N$ is the faulty signature, where S_q is changed to $S'_q \neq S_q$. We then have:

$$\begin{aligned} \Delta &\equiv S - S' \pmod{N} \\ &\equiv (aS_p + bS_q) - (aS_p + bS'_q) \pmod{N} \\ &\equiv b(S_q - S'_q) \pmod{N} \ . \end{aligned} \tag{9.15}$$

As $b \equiv 0 \pmod{p}$ and $b \equiv 1 \pmod{q}$, it follows that $\Delta \equiv 0 \pmod{p}$ (but $\Delta \not\equiv 0 \pmod{q}$) meaning that Δ is a multiple of p (but not of q). Hence, we can derive the factors of N by observing that $p = \gcd(\Delta \bmod N, N)$ and $q = N/p$.

In summary, all that is required to break RSA is one correct signature and one faulty one. This attack will be successful regardless of the type or number of faults injected during the process, provided that all faults affect the computation of either S_p or S_q.

Although initially theoretical, this attack stimulated the development of a variety of fault attacks against a wide range of cryptographic algorithms. One of the first descriptions of an implementation of this attack is given in [5].

9.5.3.2 Faults in DES

A type of cryptanalysis of ciphertext blocks produced by injecting faults into DES was proposed in [8], based on using techniques used in differential cryptanalysis [36]. One-bit faults were assumed to occur in random places throughout an execution of DES. The ciphertext blocks corresponding to faults occurring in the fourteenth and fifteenth round were taken, enabling the derivation of the key. This was possible as the effect of a one-bit fault in the last three rounds of DES is visible in the ciphertext block when it is compared with a correct ciphertext block. This allowed the key to be recovered using between 50 and 200 different ciphertext blocks. It is claimed in [8] that, if an attacker can be sure of injecting faults towards the end of the algorithm, the same results could be achieved with only ten faulty ciphertext blocks, and that, if a precise fault could be induced, only three faulty ciphertext blocks would be required.

This algorithm was improved upon in [19]. When searching for a key, the number of times a given hypothesis is found is counted. This means that faults from earlier rounds can be taken into account. It is claimed in [19] that faults from the eleventh round onwards can be used to derive information on the key, and that in the ideal situation only two faulty ciphertext blocks are required.

The simplest case of a fault attack on DES involves injecting a fault in the fifteenth round, and such an attack is well-known within the smart card industry.

The last round of DES can be expressed in the following manner:

$$\begin{aligned} R_{16} &= S(R_{15} \oplus K_{16}) \oplus L_{15} \\ &= S(L_{16} \oplus K_{16}) \oplus L_{15} \end{aligned}$$

If a fault occurs during the execution of the fifteenth round, i.e. R_{15} is randomised by a fault to become R'_{15}, then

$$\begin{aligned} R'_{16} &= S(R'_{15} \oplus K_{16}) \oplus L_{15} \\ &= S(L'_{16} \oplus K_{16}) \oplus L_{15} \end{aligned}$$

and

$$\begin{aligned} R_{16} \oplus R'_{16} &= S(L_{16} \oplus K_{16}) \oplus L_{15} \oplus S(L'_{16} \oplus K_{16}) \oplus L_{15} \\ &= S(L_{16} \oplus K_{16}) \oplus S(L'_{16} \oplus K_{16}) \ . \end{aligned}$$

This provides an equation in which only the last subkey, K_{16}, is unknown. All of the other variables are available from the ciphertext block. This equation holds for each S-box in the last round, which means that it is possible to search for key hypotheses in sets of six bits, i.e. the 48-bit output after the XOR is divided into eight groups of six bits before being substituted with values from the S-boxes.

All 64 possible key values corresponding to the XOR just before each individual S-box can be used to generate a list of possible key values for these key bits. After this, all the possible combinations of the hypotheses can be searched though, with the extra eight-key bits that are not included in the last subkey, to find the entire key.

If R'_{15} is randomised by a fault, then the expected number of hypotheses that are generated can be predicted using the methods given in [7]. Table 9.1 shows the statistically expected number of key hypotheses E_k that would be returned by a fault producing a difference across each S-box in the last round. This is an average of the non-zero elements in the expected number of hypotheses that are generated using the tables defined in [7].

Table 9.1 The expected number of hypotheses per S-box for one faulty ciphertext block

S-box	E_k
1	7.54
2	7.67
3	7.58
4	8.36
5	7.73
6	7.41
7	7.91
8	7.66

The expected number of hypotheses for the last subkey will be the product of all eight expected values E_k; this gives an expected number of around 2^{24}. This is just for the last subkey, an actual exhaustive search will need to take into account the eight bits that are not included in the last subkey, giving an overall expected keyspace size of 2^{32}.

This substantially reduces the number of possible keys that would need to be tested to try and determine the secret key used. The size of the keyspace can be further reduced if the fault attack is repeated and the intersection of the two resulting keyspaces is determined.

The same attack can also be applied if small faults occur in the last five rounds of DES, but the treatment is statistical in nature and requires many more faults to determine information on the key. Further details of this attack, and a brief description of an implementation, are given in [19]. A more detailed analysis of how DES can be attacked using faults is given in [46].

9.5.4 Countermeasures

The countermeasures that can be used to protect microprocessors from fault attacks are based on methods previously employed for integrity purposes. However, countermeasures only need to be applied in processes where an attacker could benefit from injecting a fault, although a careful analysis of a given application is required to determine where countermeasures are required. This has proven to be true even where algorithms are based on one-time random numbers, as it has been shown that the manipulation of the random number can compromise the security of a cryptographic algorithm [41]. The list of countermeasures is given below:

Checksums can be implemented in software or hardware. This prevents data (such as key values) being modified by a fault, as the fault can be detected followed by appropriate action (see below).

Execution Randomisation can be used to change the order in which operations in an algorithm are executed from one execution to another, making it difficult to predict what the machine is doing at any given cycle. For most fault attacks this countermeasure will only slow down a determined attacker, as eventually a fault will hit the desired instruction. However, this will thwart attacks that require faults in specific places or in a specific order.

Random Delays can be used to increase the time required to attack. As with execution randomisation, a determined attacker can attempt to inject a fault until the moment the fault is injected coincides with the target. However, this can take significantly more time than would otherwise be required, especially if an attacker is able to identify a target through a side channel (e.g. using Simple Power Analysis).

Execution Redundancy is the repeating of algorithms and comparing the results to verify that the correct result is generated. This is most effective when the second calculation is different to the first, e.g. the inverse function, to prevent an attacker from trying to inject an identical fault in each execution.

Variable Redundancy is the reproduction of a variable in memory. When a variable is tested or modified the redundant copy is also tested or modified. This is most effective when the copy is stored in a different form to the original, e.g. the bitwise complement, to avoid a fault being applied to each variable in the same way.

Ratification Counters and Baits can be included to prevent an attacker from successfully completing a fault attack by rendering a microprocessor inoperative once a fault attack is detected. Baits are small (<10 byte) code fragments that perform an operation and test its result. A typical bait writes, reads and compares data, performs XORs, additions, multiplications and other operations whose results can be easily checked. When a bait detects an error it increments a counter in Non-Volatile Memory (NVM), and when this counter exceeds a tolerance limit (typically three) the microprocessor ceases to function.

9.5.4.1 Remarks

Many of the countermeasures in this list can be implemented in either hardware or software. A more complete list of the countermeasures (in hardware and software), along with a description of certain fault attacks is given in [27].

9.6 Embedded Software Design

The attacks described in the previous sections of this chapter have focused on attacking cryptographic algorithms to determine a secret or private key. In this section some example of how the attack methods presented in Sects. 9.4 and 9.5 can be applied to other security mechanisms are described. This is to demonstrate that implementing a secure application on a smart card is not trivial, and requires the careful evaluation of every implemented function. It should also be noted that the attacks described below are only possible where no specific countermeasures are implemented.

9.6.1 PIN Verification

As described in Sect. 9.3, the first smart cards included a contact that was called Vpp used to supply power to the microprocessor so that it could program the EEPROM present in the chip. At the time, the voltage supply (Vcc) did not supply enough power to allow a microprocessor to modify EEPROM and a higher voltage needed to be applied to the Vpp contact. The Vpp contact is no longer used as it led to security problems, as described below.

If the Vpp contact was masked (e.g. covered with nail varnish) then no power would be available for the microprocessor to program the EEPROM, but power would be available through the Vcc to run every other function of the smart card. This meant that an attacker could try every single PIN number without decrementing the PIN counter (typically a PIN counter is set to three and decremented with every false PIN presentation, once the PIN number is zero a smart card will render itself non-functional). This process could be automated using a standard PC and a smart card reader to determine a PIN number in matter of minutes.

After the Vpp contact was removed further problems were encountered. The most natural way to implement a PIN verification would be as shown in Algorithm 3.

Algorithm 3: Insecure PIN Verification Algorithm

```
Input: PIN
Output: Whether the PIN is valid, true or false.
if (PINcounter > 0) then
    PIN ← RequestPIN();
else
    return false ;
end
if PIN ≠ UserPIN then
    PINcounter ← PINcounter − 1 ;
    return false ;
else
    return true ;
end
```

The PIN entered by a user is returned by the RequestPIN function and compared with the PIN in Non-Volatile Memory (NVM). If the entered PIN is not equal to the stored PIN, the PINcounter will be decremented.

It was observed that the power consumption increased when a smart card modified the value of the PIN counter, i.e. it was visible using the SPA techniques described in Sect. 9.4 as an increase in the power consumption. Attackers then developed tools to cut the power being supplied to a microprocessor once this increase in power consumption was detected. This allowed automated tools to attempt every PIN number until the correct PIN was found. The correct PIN would be the only value where the command would finish as the microprocessor would not attempt to modify the NVM.

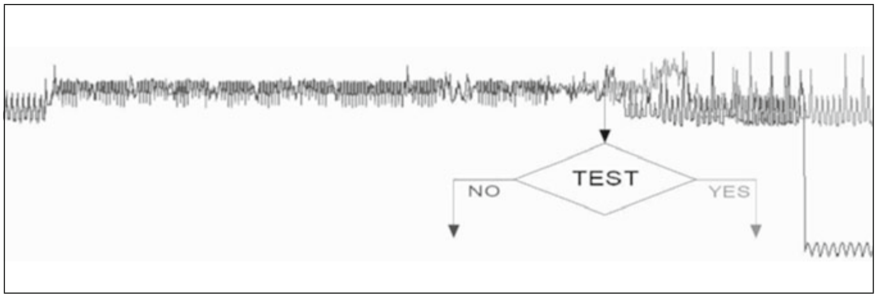
This made it necessary to change the algorithm used to verify a PIN number. Typically, a secure PIN verification will be implemented in the following manner: where the PINcounter is decremented before it is tested, and only incremented if the PIN is entered correctly. The power supply can be removed at any point during the command without producing a security problem. However, further modifications would need to be made to render it resistant to fault attacks. An example of a fault attack against an operating system is described in Algorithm 4.

Algorithm 4: Secure PIN Verification Algorithm

```
Input: PIN
Output: Whether the PIN is valid, true or false.
if (PINcounter > 0) then
    PIN ← RequestPIN() ;
else
    return false ;
end
PINcounter ← PINcounter − 1; if (PIN = UserPIN) then
    PINcounter ← PINcounter + 1; return true ;
else
    return false ;
end
```

9.6.2 *File Access*

Another possible target within a smart card operating system is the file structure. All personalisation information, e.g. PIN numbers etc., is stored in a file structure situated in NVM. Each file will have a set of access conditions that determine who can read and write to each file or directory. For example a user's PIN number on a SIM card, unless intentionally disabled, will grant access to the files that contain SMS messages once verified. If the PIN is not verified access to these files will be denied. There are often administrator identification codes (essentially eight digit PIN numbers) that grant access to more files and allow the modification of files that the end user is not able to directly modify, e.g. the user's PIN number.



Key: **Black** Power consumption where file access is denied.
Grey Power consumption where file access is granted.

Fig. 9.13 Determining the moment file access is granted using the power consumption

If an attacker wishes to attempt to access information stored in files without any of the codes mentioned above, a fault attack could be attempted. An attacker can attempt to inject a fault at the moment a smart card is evaluating whether the right to read a file, for example, should be granted. If successful, the evaluation will be erroneous and the right to access the file will be temporarily granted.

In order to determine the point in which a fault would need to be injected, an attacker can use SPA (see Sect. 9.4). An attacker could compare a trace of the power consumption where file access is granted with a trace where file access has been denied. An example of this is shown in Fig. 9.13. The black trace represents the power consumption where access has been denied. The grey trace represents the power consumption where access has been granted. It can be see at the point indicated in the figure that the two traces diverge. This should represent the moment at which the access conditions are evaluated and will, therefore, be the targeted area for a fault attack.

On a smart card there are typically files that contain serial numbers and such information, which can be read by anyone. Finding two files to attempt to read in order to generate traces, as shown in Fig. 9.13, should be straightforward.

This type of fault attack means that the access conditions to files, and other security mechanisms such as PIN verification, need to include redundancy in their tests to ensure that a fault attack is not possible. The various countermeasures that can be implemented are described in Sect. 9.5.

9.7 In Conclusion

This chapter presents the particular security considerations that need to be taken into account when implementing a secure smart card-based application. Implementations of all the commands on a smart card need to be subjected to careful analysis to prevent power analysis and fault injection techniques from compromising the security of the smart card.

Research in the domain of smart card security is typically a cyclic process. New attacks are developed against algorithm implementations, standards, etc. and countermeasures are proposed. The modifications are then reviewed for potential vulnerabilities and further countermeasures proposed if required. The aim of this process is to remain sufficiently ahead of what can be achieved by an individual attacker that smart cards remain secure throughout the period that they are active.

References

1. Akkar, M.-L. and Giraud, C. (2001). An implementation of DES and AES secure against some attacks. In Koç, C. K., Naccache, D., and Paar, C., editors, *Cryptographic Hardware and Embedded Systems—CHES 2001,* volume 2162 of *Lecture Notes in Computer Science,* pages 309–318. Springer-Verlag.
2. American National Standards Institute (1985). *Financial Institution Key Management (Wholesale).* American National Standards Institute.
3. Anderson, R. and Kuhn, M. (1996). Tamper resistance—a cautionary note. In *Proceedings of the Second USENIX Workshop of Electronic Commerce,* pages 1–11.
4. Anderson, R. and Kuhn, M. (1997). Low cost attacks on tamper resistant devices. In Christianson, B., Crispo, B., Lomas, T. M. A., and Roe, M., editors, *Security Protocols,* volume 1361 of *Lecture Notes in Computer Science*, pages 125–136. Springer-Verlag.
5. Aumüller, C., Bier, P., Hofreiter, P., Fischer, W., and Seifert, J.-P. (2002). Fault attacks on RSA with CRT: Concrete results and practical countermeasures. In Kaliski, B. S., Koç, C. K., and Paar, C., editors, *Cryptographic Hardware and Embedded Systems—CHES 2002,* volume 2523 of *Lecture Notes in Computer Science,* pages 260–275. Springer-Verlag.
6. Bar-El, H., Choukri, H., Naccache, D., Tunstall, M., and Whelan, C. (2006). The sorcerer's apprentice guide to fault attacks. *Proceedings of the IEEE,* 94(2):370–382.
7. Biham, E. and Shamir, A. (1991). Differential cryptanalysis of DES-like cryptosystems. In Menezes, A. and Vanstone, S., editors, *Advances in Cryptology—CRYPTO '90,* volume 537 of *Lecture Notes in Computer Science,* pages 2?-21. Springer-Verlag.
8. Biham, E. and Shamir, A. (1997). Differential fault analysis of secret key cryptosystems. In Kaliski, B. S., editor, *Advances in Cryptology—CRYPTO '97,* volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer-Verlag.
9. Blömer, J. and Seifert, J.-P. (2003). Fault based cryptanalysis of the advanced encryption standard (AES). In Wright, R. N., editor, *Financial Cryptography—FC 2003,* volume 2742 of *Lecture Notes in Computer Science*, pages 162–181. Springer-Verlag.
10. Boneh, D., DeMillo, R. A., and Lipton, R. J. (1997). On the importance of checking computations. In Fumy, W., editor, *Advances in Cryptology—EUROCRYPT '97,* volume 1233 of *Lecture Notes in Computer Science,* pages 37–51. Springer-Verlag.
11. Brier, E., Clavier, C., and Olivier, F. (2004). Correlation power analysis with a leakage model. In Joye, M. and Quisquater, J.-J., editors, *Cryptographic Hardware and Embedded Systems—CHES 2004,* volume 3156 of *Lecture Notes in Computer Science*, pages 16–29. Springer-Verlag.
12. Chari, S., Jutla, C. S., Rao, J. R., and Rohatgi, P. (1999). Towards approaches to counteract power-analysis attacks. In Wiener, M., editor, *Advances in Cryptology—CRYPTO '99,* volume 1666 of *Lecture Notes in Computer Science,* pages 398–412. Springer-Verlag.
13. Chevallier-Mames, B., Ciet, M., and Joye, M. (2004). Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity. *IEEE Transactions on Computers,* 53(6):760–768.
14. Clavier, C., Coron, J.-S., and Dabbous, N. (2000). Differential power analysis in the presence of hardware countermeasures. In Koç, C. K. and Paar, C., editors, *Cryptographic Hardware*

and Embedded Systems—CHES 2000, volume 1965 of *Lecture Notes in Computer Science,* pages 252–263. Springer-Verlag.
15. Dobraunig, C., Eichlseder, M., Mangard, S. and Mendel, F. (2014). On the Security of Fresh Re-keying to Counteract Side-Channel and Fault Attacks. In Joye, M. and Moradi, A., editors, *Smart Card Research and Advanced Applications—13th International Conference, CARDIS 2014*, volume 8968 of *Lecture Notes in Computer Science*, pages 233–244. Springer-Verlag.
16. Fouillat, P. (1990). *Contribution à l'étude de l'interaction entre un faisceau laser et un milieu semiconducteur, Applications à l'étude du Latchup et à l'analyse d'états logiques dans les circuits intégrés en technologie CMOS.* PhD thesis, University of Bordeaux.
17. Gandolfi, K., Mourtel, C., and Olivier, F. (2001). Electromagnetic analysis: Concrete results. In Koç, C. K., Naccache, D., and Paar, C., editors, *Cryptographic Hardware and Embedded Systems—CHES 2001,* volume 2162 of *Lecture Notes in Computer Science,* pages 251–261. Springer-Verlag.
18. Genkin, D., Pachmanov, L., Pipman, I., Tromer, E. (2015). Stealing Keys from PCs Using a Radio: Cheap Electromagnetic Attacks on Windowed Exponentiation. In Güneysu, G. and Handschuh, H., editors, *Cryptographic Hardware and Embedded Systems—CHES 2015,* volume 9293 of *Lecture Notes in Computer Science,* pages 207–228. Springer-Verlag.
19. Giraud, C. and Thiebeauld, H. (2004). A survey on fault attacks. In Deswarte, Y. and Kalam, A. A. El, editors, *Smart Card Research and Advanced Applications VI—18th IFIP World Computer Congress,* pages 159–176. Kluwer Academic.
20. Goodwill, G., Jun, B., Jaffe, J. and Rohatgi, P. (2011). A testing methodology for side-channel resistance validation. In *The Non-Invasive Attack Testing Workshop—NIAT 2011.*
21. Govindavajhala, S. and Appel, A. W. (2003). Using memory errors to attack a virtual machine. In *IEEE Symposium on Security and Privacy 2003,* pages 154–165.
22. Gutmann, P. (2004). *Security Architecture.* Springer.
23. Habing, D. H. (1992). The use of lasers to simulate radiation-induced transients in semiconductor devices and circuits. *IEEE Transactions On Nuclear Science,* 39:1647–1653.
24. International Organization for Standardization (1997). *ISO/IEC 7816–3 Information technology—Identification cards—Integrated circuit(s) cards with contacts – Part 3: Electronic signals and transmission protocols.* International Organization for Standardization.
25. International Organization for Standardization (1999). *ISO/IEC 7816–2 Identification cards—Integrated circuit cards—Part 2: Cards with contacts—Dimensions and location of the contacts.* International Organization for Standardization.
26. Joye, M. and Olivier, F. (2005). Side-channel attacks. In van Tilborg, H., editor, *Encyclopedia of Cryptography and Security,* pages 571–576. Kluwer Academic Publishers.
27. Joye, M. and Tunstall, M., Eds (2015). Fault Analysis in Cryptography. Springer.
28. Kahn, D. (1997). *The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet.* Simon & Schuster Inc., second edition.
29. Knuth, D. (2001). *The Art of Computer Programming,* volume 2, Seminumerical Algorithms. Addison–Wesley, third edition.
30. Kocher, P. (1996). Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Koblitz, N., editor, *Advances in Cryptology—CRYPTO '96,* volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer-Verlag.
31. Kocher, P., Jaffe, J., and Jun, B. (1999). Differential power analysis. In Wiener, M. J., editor, *Advances in Cryptology—CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer-Verlag.
32. Kommerling, O. and Kuhn, M. (1999). Design principles for tamper resistant smartcard processors. In *USENIX Workshop on Smartcard Technology,* pages 9–20.
33. Longo Galea, J., De Mulder, E., Page, D. and Tunstall, M. (2015). SoC It to EM: ElectroMagnetic Side-Channel Attacks on a Complex System-on-Chip. In Güneysu, G. and Handschuh, H., editors, *Cryptographic Hardware and Embedded Systems—CHES 2015,* volume 9293 of *Lecture Notes in Computer Science,* pages 620–640. Springer-Verlag.
34. Mangard, S., Oswald, E., and Popp, T. (2007). *Power Analysis Attacks—Revealing the Secrets of Smart Cards.* Springer-Verlag.

35. May, T. and Woods, M. (1978). A new physical mechanism for soft errors in dynamic memories. In 16*th* International Reliability Physics Symposium.
36. Menezes, A., van Oorschot, P., and Vanstone, S. (1997). *Handbook of Applied Cryptography.* CRC Press.
37. Messerges, T. S. (2000). *Power Analysis Attacks and Countermeasures for Cryptographic Algorithms.* PhD thesis, University of Illinois, Chicago.
38. Meyer, C. (2000). Private communication. Carl Meyer was one of the designers of the DES algorithm.
39. MIPS-Technologies (2001). MIPS^{TM} architecture for programmers volume I: Introduction to the MIPS32^{TM} architecture. Technical Report MD00082, Revision 0.95.
40. Murdocca, M. and Heuring, V. P. (2000). *Principles of Computer Architecture.* Addison-Wesley.
41. Naccache, D., Nguyen, P. Q., Tunstall, M., and Whelan, C. (2005). Experimenting with faults, lattices and the DSA. In Vaudenay, S., editor, *Public Key Cryptography—PKC 2005,* volume 3386 of *Lecture Notes in Computer Science,* pages 16–28. Springer-Verlag.
42. NIST (1999). *Data Encryption Standard (DES) (FIPS-46–3).* National Institute of Standards and Technology.
43. NIST (2001). *Advanced Encryption Standard (AES) (FIPS-197).* National Institute of Standards and Technology.
44. Pouget, V. (2000). *Simulation expérimentale par impulsions laser ultra-courtes des effets des radiations ionisantes sur les circuits intégrés.* PhD thesis, University of Bordeaux.
45. Quisquater, J.-J. and Samyde, D. (2001). Electromagnetic analysis (EMA): Measures and counter-measures for smart cards. In Attali, I. and Jensen, T. P., editors, *Smart Card Programming and Security, International Conference on Research in Smart Cards—E-smart 2001,* volume 2140 of *Lecture Notes in Computer Science,* pages 200–210. Springer-Verlag.
46. Rivain, M. (2009). Differential Fault Analysis on DES Middle Rounds. Clavier, C. and Kris, G., editors, *Cryptographic Hardware and Embedded Systems—CHES 2009,* volume 5747 of *Lecture Notes in Computer Science,* pages 457–469. Springer-Verlag.
47. Rivest, R., Shamir, A., and Adleman, L. M. (1978). Method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM,* 21(2):120–126.
48. Samyde, D., Skorobogatov, S. P., Anderson, R. J., and Quisquater, J.-J. (2002). On a new way to read data from memory. In *Proceedings of the First International IEEE Security in Storage Workshop,* pages 65–69.
49. Schneider, T. and Moradi, A. (2015). Leakage Assessment Methodology—A Clear Roadmap for Side-Channel Evaluations. In Güneysu, G. and Handschuh, H., editors, *Cryptographic Hardware and Embedded Systems—CHES 2015,* volume 9293 of *Lecture Notes in Computer Science,* pages 495–513. Springer-Verlag.
50. Skorobogatov, S. and Anderson, R. (2002). Optical fault induction attacks. In Kaliski, B. S., Ç. K. Koç, and Paar, C., editors, *Cryptographic Hardware and Embedded Systems—CHES 2002,* volume 2523 of *Lecture Notes in Computer Science,* pages 2–12. Springer-Verlag.
51. Skorobogatov, S. P. (2005). *Semi-Invasive Attacks—A New Approach to Hardware Security Analysis.* PhD thesis, University of Cambridge. available at http://www.cl.cam.ac.uk/TechReports/.
52. Tarnovsky, C., (2015) https://en.wikipedia.org/wiki/Christopher_Tarnovsky. Accessed 16 November 2015.
53. Wright, P. (1987). *Spycatcher.* Heineman.
54. Ziegler, J. (1979). Effect of cosmic rays on computer memories. *Science,* 206:776–788.

Chapter 10
Application Development Environments for Java and SIM Toolkit

Gary Waite, Keith Mayes and Raja Naeem Akram

Abstract The smart card is a very popular component of many commercial and government system solutions. The ability of the smart card to store data securely and resist a great deal of physical tampering is part of the attraction, but so too is the ability to run algorithms and protocols. Whilst, there are successful and popular systems that make use of fairly simple cards with fixed function algorithms, the true potential of the smart card can only be realised when it represents a flexible platform for general application hosting and management. Fortunately, such functionality is becoming commonplace on modern cards and so the focus moves to how applications may be practically developed to exploit it. There are a range of ways this can be done, but by way of illustration this chapter restricts itself to Java as one of the most popular development methods and applies it to (Universal) Subscriber Identity Modules ((U)SIMs); perhaps the most powerful of the mass deployed smart cards.

Keywords Development · API · Java · Java card · Java SIM · Java framework · SIM toolkit · SATSA · JSR177 · Applet · Midlet · Events · Tools · Testing · DRM · Software

G. Waite
Executive Director—Embedded SIM, GSMA, London, UK
e-mail: me@garywaite.com

K. Mayes
Director of the Information Security Group, Head of the School of Mathematics and Information Security, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK
e-mail: keith.mayes@rhul.ac.uk

R.N. Akram (✉)
Smart Card Centre, Information Security Group, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK
e-mail: r.n.akram@rhul.ac.uk

K. Mayes and K. Markantonakis (eds.), *Smart Cards, Tokens, Security and Applications*, DOI 10.1007/978-3-319-50500-8_10

10.1 Introduction

A modern state-of-the-art smart card is a powerful multi-application platform incorporating sophisticated and secure management functionality. This enables applications to be added, deleted or modified both during the pre-issuance and post-issuance phases. Indeed, it is possible to have a sophisticated hierarchy of security rights and permissions that delegate management authority to third parties. Most of the important features of smart card platforms have been well standardised, but the application platform has evolved into a few variants, as discussed in Chap. 3. Similarly there are different ways of designing smart card applications such as the browser approach mentioned in Chap. 1. In a single book chapter, it is not possible to go into detail on every possible variant of platform, application and development method. Choosing a platform to focus on is not difficult as Java Card [1] is a popular de-facto standard and the Java language and tools are familiar to developers. Selecting the card type is a little more open, but as discussed in Chap. 4, the Subscriber Identity Module (SIM) [2] is a good choice, for the reason that it is one of the most advanced smart cards in widespread use. This chapter will therefore focus on application development for Java Card-based SIMs [2], but as a first step it will revisit some of the fundamental characteristics of the smart card that make it interesting as an application host. During the discussion regarding the Java Card development, we will take into consideration both the Java Card 2.x.x and 3.x.x variants, along with providing a quick and short guide on how to program for them. Note that, where there is no need to differentiate between a SIM and a 3G/4G USIM, the terms SIM will be used for the device and SIM Toolkit for the Toolkit capabilities.

10.2 Smart Cards Characteristics

The experienced software developer, used to personal computers (PCs) and giant servers might wonder what makes the concept of a smart card so special for use in development? Well, there are in fact a number of useful attributes that normally go unnoticed or are simply taken for granted. First, a smart card is small, flat and portable. This in itself may not seem particularly special, but it is when you consider that all of its functionality is contained in a flat piece of plastic that can fit inside a wallet and be carried around to interact with a wide range of devices and readers. It is also a secure hardware platform, which essentially means that although the smart card may hold sensitive data it will not divulge this to just anyone (or any machine). The data is protected by sophisticated measures within the chip and various security policies, from Personal Identification Number (PIN), to mutual authentication challenges. Of course attacks are attempted against smart card security (as described in Chap. 9, but unless something has been badly designed or implemented, the associated time, effort cost and expertise are not usually merited by the value of the data discovered, meaning that the economic benefit of the attack is far less then the investment an attacker has to make. The smart card also benefits from a global standard interface [3], which in real terms means that it will inter-operate with most terminal equipment around the world. This is an important point, as so many technologies coming to market these days promise powerful and compelling services, but fail due to lack of a widely agreed standard interface. Smart cards are also very reliable, which is a vital attribute for any deployed or issued device. They seem to work way beyond their warranted lifespan and can take all manner of abuse including being dropped from great heights, repeatedly sat on, withstanding low and high temperatures, surviving duty as a windscreen ice scraper and even being submerged in beer! Whilst some cards, such as credit cards have a controlled lifetime before re-issue like bank cards, SIM cards do not and there are still many in regular use that are potentially over 10 years old. Smart cards can also be branded, for example with the issuing organisation. This may not seem particularly noteworthy especially in a chapter about application development, but branding is a hot issue even for single application cards and so one can imagine the difficulties with multi-application cards, especially if the applications are not all loaded or even known pre-issuance. The Issuer of a smart card, e.g. a bank or network operator, is in a prime position to control the branding and indeed general functionality of the issued card, however the influence and control does not stop there. Often in the fine print of a user agreement you will find that ownership of the card remains with the Issuer and that the user is merely a temporary custodian. This means that the Issuer can manage, change and generally control the card post-issuance, using secure mechanisms that are usually denied to other parties. Having a smart card, in your wallet or mobile phone makes it very personal, especially as you have a measure of control over stored data and functionality, by means of your user PIN code. This brings us to an often forgotten attribute of a smart card in that despite its role in security and sensitive data handling it is not regarded as a threatening technology. Because smart cards have been in use by the general public for many years (moreover: used by the general public for trusted services), and because they're small and flat, people are not too worried about having or using them. The user is ultimately in charge in a non-technical sense, as a pair of scissors can easily bring about the demise of a smart card, if required, which reminds us of another attribute; that smart cards are disposable. In the twenty-first century, disposability and associated recycling are major concerns, however smart cards have a low waste factor and being essentially plastic can be recycled by the manufacturer. Finally, smart cards are relatively cheap, meaning that they can provide a cost effective solution for many services today.

With all these desirable attributes one might imagine that the smart card platform is a software developer's dream, but alas that is not really the case as any application is constrained to work within a number of practical limitations described in the next section.

10.2.1 Limitations

Smart cards have very limited memory, typically anywhere between 16 and 256 Kb. You can buy cards with multi-Megabyte an even Gigabyte memory, but these are expensive and the technology is still being rolled out. So for at least, any applications requiring vast amounts of local smart card memory are impractical.

Smart cards also have limited processing power. This is partly due to the size and vintage of the CPUs, but also to the power that they allowed to draw from a reader device. Generally, they are inappropriate for complex number-crunching or bulk data processing with the exception of cryptographic functions, where the chip may have a specialised co-processor for rapid execution. They also have limited Application Programming Interfaces (API)s, which in turn limit the creativity and flexibility for application developers. There is further disappointment as current smart cards have limited connectivity. Clearly, there is no connectivity without a reader, but even when the reader is present, the connectivity is severely limited by the card-reader interface.

The very nature of the smart card being a secure device, means that the opportunities for developers are generally limited to those linked to the card issuing organisations. This is because the security keys used to manage the device are generally not known to third parties. So even though the Java Card Open Platform (the most popular development environment) is a global standard, and specifications are freely available to any developer, practical open-access to such cards is difficult, if not impossible without permission by the Card Issuers.

Once cards are eventually deployed with one or more useful applications, there is limited opportunity for post-issuance management of the applications. Limited does not mean impossible and there are standardised protocols designed specifically to manage and/or change data and applications, however practical considerations often restrict management to minor changes. For example, in mobile-related applications there is the concept of over the Air (OTA) management (see Chap. 11), but it is heavily bandwidth-restricted; mainly because of the SMS-data bearer used. Furthermore the sheer number of cards deployed makes any kind of management difficult. For example, managing and maintaining the upgrade state of millions of smart cards of differing ages and capabilities, and when they may only be able to communicate at certain times and without annoying the user, is really quite a challenge.

Finally, one should not expect the latest and greatest smart card features and standards to be implemented overnight. Aside from the time to rollout new cards, the standards themselves tend to be slow to ratify new capabilities. It should also be noted that this is not simply due to technical debate, but also due to political and IPR (Intellectual Property Right) interests. Indeed in some markets, including mobile communications, there is great pressure applied by some companies to limit what smart cards can do.

10.3 SIM Cards

SIM and USIM cards were introduced in Chap. 1 and described in detail within Chap. 4, so here we will just recap on some of the application relevant capabilities. A SIM card is essentially a smart card running at least a GSM application. For developers the main API is provided by a technology called SIM Application Toolkit (SAT or STK), but generalised for all smart cards under the name of Card Application Toolkit (CAT). (Note the terms CAT, SAT and STK may be used interchangeably from now on)

CAT allows applications running on the SIM card to access useful services on the mobile handset. These include the ability to send a short message, operate a simple User Interface (UI) with the handset, or perhaps to retrieve the handset's current location.

Some examples of deployed SIM applications are:

IMEI trigger—it is very useful for the operator to know the particular handset that is being used by each customer (for example to ensure the correct GPRS/WAP/MMS settings are sent). An application on the SIM is able to retrieve the International Mobile Equipment Identity (IMEI) from the handset when the SIM is powered on, then compare it with a list of IMEIs stored on the SIM. If the IMEI is new (i.e. not in the list), the operator is automatically notified via SMS, so that appropriate settings can be sent to the handset.

Preferred network selection—when roaming, some operators like to steer their customers to particular networks and this can be achieved using a SIM application.

Digital rights management—in the same way that Universal Serial Bus (USB) dongles can be used to copy-protect software running on PCs, the SIM can also be employed for a similar purpose.

The listed applications are only a small sub-set of what is possible and indeed in regular use today. The reader may have many other ideas for services and so an important question is how to write applications that can run on a SIM. When considering multi-application SIMs we are normally talking about a Java Card platform running a SIM application and so we first need to recap on the that platform.

10.4 Java Card

In this section, we discuss the recent Java Card specification 3 that has two distinct editions: Java Card Classic and Connected. The following discussion follows the material presented in [4].

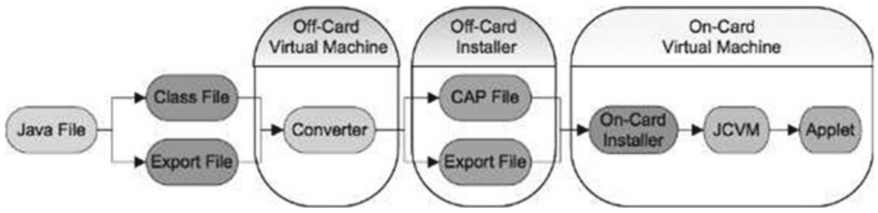


Fig. 10.1 Java Card Classic edition application development cycle

10.4.1 Java Card Classic

The Java Card Classic edition is an incremental evolution to the Java Card 2.2.2 [5]. It maintains backward compatibility and designed for resource restricted devices. The Java Card Classic virtual machine is similar to the previous Java Card virtual machines that had a split architecture. This architecture was divided between on-card and off-card processing of an application. Before loading an application on to a Java Card, the application code is analysed by an off-card virtual machine to verify its integrity and compliance with Java Card specification. This analysed (pre-processed) application is then loaded onto a Java Card. The Java Card 3 Classic fixes some bugs, provide clarifications to the Java Card 2.2.2, along with providing support for security algorithms like 4096-bit RSA and NSA Suite B [6] and minor improvements on how contactless transactions are processed.

A typical Java Card 3 Classic edition and Java Card 2.2.2 supports the following features:

1. Only support boolean, byte and short data types.
2. Single thread execution.
3. Garbage collection (best effort).
4. Single dimension arrays.
5. Exceptions.
6. Support Java Card Remote Method Invocation (RMI), and ISO/IEC 7816-4 (e.g. APDU) for off-card communication.
7. Dedicated API to support biometric data.

10.4.1.1 Java Card Classic Application Development Cycle

The traditional Java Card application development cycle is illustrated in Fig. 10.1. After coding an application, a developer will compile the corresponding Java file to produce class and export files.

The class file contains the application related bytecode and export file has any associated configurations. These two files are then converted by the off-card virtual machine to a CAP[1] file and associated export file. The off-card virtual machine also performs the bytecode verification to check whether the application conforms with the Java Card specification.

After the generation of the CAP and export file by the off-card virtual machine, these files are input to the off-card installer that communicates them to an on-card installer. The on-card installer downloads the application code on to the respective smart card. Once the application is downloaded it registers itself with the JCVM. The application can execute and communicate with on-card or off-card entities. In most of the latest smart cards that support Java Card specification, GlobalPlatform specification[2] [5] is used for off-card and on-card installers, along with post-issuance application management.

10.4.2 Java Card Connected

The Java Card Connected edition is being proposed for high-end smart cards. It contains a scaled down version of Java's Connected Limited Device Configuration (CLDC) virtual machine that is used in the mobile phones. Therefore, the Java Card Connected can support the feature rich programming environment provided by the CLDC virtual machines. It also enables a smart card to communicate concurrently on variety of interfaces: using Internet Protocol (IP), ISO/IEC 7816-4 protocols, and contactless interfaces (ISO/IEC 14443 [7]).

The Java Card Connected edition moves the smart card into the realm of powerful network node that is capable of acting as an independent platform providing security and Privacy services. Therefore, Java Card Connected edition may be the road ahead for the smart card technology. The features provided by it are listed below:

1. Support for all data types except for float and double.
2. Enable multithreading.
3. Support feature rich APIs like Generic Connection Framework (GCF), servlet, sockets, threads, and transactions, etc.
4. On-card bytecode verification.
5. Automatic garbage collection.
6. Multidimensional arrays.
7. Support primitive wrapper classes (e.g. Boolean and Integer), string manipulation classes (e.g. StringBuffer, StringBuilder, etc.), Input/Output (I/O) classes (e.g. Reader, Writer, and Stream), and collection classes (e.g. Vector, Hashtable, Stack, and Iterator, etc.).

[1]CAP: The Java Card **C**onverted **AP**plet (CAP) is an Java Card interoperable file format used to deploy an application on smart cards.

[2]GlobalPlatform: The GlobalPlatform specification provides a secure, reliable and interoperable application management framework for a multi-application smart cards. In this framework, individual application providers can manage their applications in a secure and reliable manner without relying on the respective card issuer.

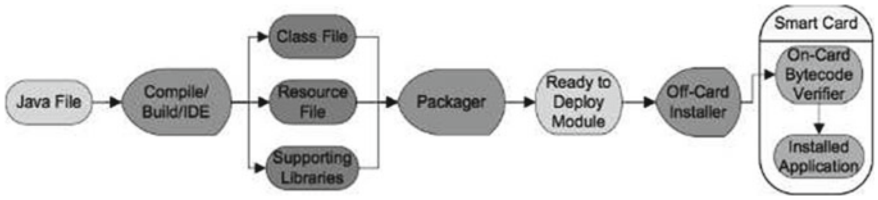


Fig. 10.2 Java card connected application development cycle

8. It also supports Java SE features like generics, metadata, assertions, enhanced for loop, varargs and static inputs, etc.

In addition to these features, the Java Card Connected edition supports the loading of a Java application from Java Archive (JAR) file.

10.4.2.1 Java Card Connected Application Development Cycle

The Java Card Connected application development cycle is illustrated in Fig. 10.2. The application code is compiled to produce class and resource files along with supporting libraries. The Java application does not have to be in the CAP file format for the Java Card Connected edition, as it supports JAR format. Therefore, a packager will take the output of the compiler (e.g. supporting libraries, class and resource files) and packages them into a module (it can be either CAP or JAR depending upon the smart card's supported format) that is ready to be deployed. This module is then input to the off-card installer that loads the application to the respective smart card. As discussed before the Java Card Connected has an on-card bytecode verifier; therefore, there is no particular need to perform off-card byte code verification.

Although, the application loading is different for the Java Card Classic and Connected editions, most of the modern integrated development environments (IDEs) can hide these details from the developer. Examples of the IDE for Java Cards are NetBeans and Eclipse. A developer can use other IDEs, but we choose these two for this chapter.

10.5 Java Card Programming

In this section, we begin with a short description of the traditional Java Card applet supported by Java Card version 2.2.2 and 3.1 Classic Edition. This will give an understanding of the basic structure of a Java Card applet.

10.5.1 Java Card Applet Architecture

The basic required structure of a Java Card applet is shown in Listing 10.1. Every applet in the Java Card has to be associated with a package. A package can have multiple applets and it can be considered as a container on a smart card that has several components of the respective application.

```
1  package myFirstApplet;
2
3  import javacard.framework.APDU;
4  import javacard.framework.Applet;
5  import javacard.framework.ISOException;
6
7  public class FirstApplet extends Applet {
8
9      private FirstApplet() {
10     }
11
12     public static void install(byte bArray[], short bOffset, byte
           bLength)
13       throws ISOException {
14         new FirstApplet().register();
15     }
16
17     public void process(APDU apduHandle) throws ISOException {
18
19     }
20 }
```

Listing 10.1 Basic Architecture of a Java Card Applet

The *import* statements includes the Java Card API that a developer may want to utilise in his/ her applet. For brevity, we do not dive into the discussion of the entire Java Card API. The first *importjavacard.framework.APDU* statement includes the Application Protocol Data Unit (APDU) support to the applet, which is necessary if the developer requires his/ her applet to communicate with off-card entities using APDUs. Similarly, developers also have to import the *javacard.framework.Applet* from which all Java Card applets have to be derived. The *install* and *process* methods listed at lines 12 and 17 are extended from the *javacard.framework.Applet*. The reasons for implementing these methods in the class is described later in this section. Finally, *javacard.framework.ISOException* is included to enable the applet to handle and signal exceptions. This is a useful feature for the robust design of an applet that if used properly can manage any possible errors during the application execution.

The two most important methods that a Java Card applet has to implement are: *install* and *process*. The *install* method is called by the Java Card Runtime Environment (JCRE) to create an instance of the Applet subclass. In a simplistic manner, it can be explained as registering an applet with the JCRE. Once an applet is registered, it becomes selectable to an off-card entity enabling it to communicate with the applet . The *install* method is invoked at the end of the application loading, and usually it is requested by the off-card installer. If JCRE do not execute the *install* method, then the application is loaded on the smart card but it would not be able to communicate with other entities (on-card or/and off-card). The *install* method accepts a byte array as an input parameter that can be sent by the off-card installer to personalise an application. The data items that a developer wants to execute only once in the lifetime of an applet should be declared in this method, examples of such data structures can be cryptographic keys associated with the application and any application personalisation information.

The *process* method is invoked by the JCRE, when an off-card entity sends an APDU for the applet. The input parameter of the process method is the APDU object which contains the byte array containing the APDU sent by the off-card entity. This method can be considered as the main function of the applet as it acts an interface between an off-card entity and the functionality implemented by the respective applet. Depending upon the design of the application, the applet will perform the required function.

The JCRE comprises Java Card Virtual Machine (VM) and classes within the Java Card Framework. Each applet on the card has unique Application Identifier (AID) assigned. The "install()" method (if successful) creates an instance of the application and registers it with the JCRE. It is generally expected that the developer will create all objects to be used during the life of the applet at installation. An Applet becomes active on a SELECT APDU command from the terminal. Any previously selected applet is immediately suspended (with its "deselect()" method being called so that the applet is informed), and the newly active applet has its "select()" method called. All APDU commands sent to the card (other than a SELECT command to select another applet) are routed through to the active applet where they are handled by the "process()" method of the selected applet.

10.5.2 Java Card Applet—Hello World

The example shown in Listing 10.2 is the equivalent of the "Hello World" program in conventional programming. The class supports the install method which effectively creates the new applet instance. The instance is registered with the JCRE and becomes available for use. The most interesting method for a developer is the "process (APDU)" as this is where all the message handling takes place. The message handler can decide what action is appropriate by examining a buffer containing the incoming APDU. The examination normally takes place in a structured manner, first checking that the APDU class byte is appropriate for the applet and then determining an appropriate handler based on the instruction byte. Individual command handlers would normally be implemented in separate sub-routines where more precise parameter checking would take place to determine the appropriate action and response.

```
package com.o2.JavaCard;

import javacard.framework.*;
import javacardx.framework.*;

public class MyFirstJavaCardApplet extends Applet {
  byte[] buffer;

    private MyFirstJavaCardApplet() {
       register();
    }

    public static void install(byte bArray[], short bOffset, byte
         bLength)
      throws ISOException {
        new MyFirstJavaCardApplet();
    }

    public void process(APDU apduHandle) throws ISOException {
    buffer = apdu.getBuffer();

        if(buffer[ISO.OFFSET_CLA] != (byte)0x80)
          ISOException.throwIt(ISO.SW_CLA_NOT_SUPPORTED);

        if(buffer[ISO.OFFSET_INS] != (byte)0x99)
          ISOException.throwIt(ISO.SW_INS_NOT_SUPPORTED);

          // Process command here!
    }
}
```

Listing 10.2 A Simple Java Card applet

A pre-requisite to examining the APDU fields, is to know how to navigate the message buffer array. As buffers could be implemented in a variety of ways the particular message bytes are found using defined "ISO" indexes. For example in Listing 10.2, the index "ISO.OFFSET_CLA" is used to index the buffer array position where the APDU class byte has been stored. In the simple program example only commands are accepted with a certain class byte "(80)" and even then only commands with a certain instruction byte "(99)". Upon receipt of a valid command the applet completed normally and a good status "(90 00)" is send back to the terminal. Although this is a simple example, simplicity is not something that can be forgotten when developing software for a Java Card, even when fleshing out a fully functional application.

Having knowledge of general Java programming is of course an advantage for anyone developing for Java Card; however it may come as a surprise that many of the classic Java features are to be avoided. Probably, the first thing that a developer would notice looking at commercial-grade Java Card applet source code, is the complete absence of object-orientation! Although one of the foundations of Java, object orientation is not really used for Java Card, because memory is at such a premium. Object orientation on small platforms is often more trouble than it is worth and developers focus instead on trying to fit into one byte what they would normally have written an entire class for. Java Card developers also use the "new" keyword sparingly or not at all as an object created can never be destroyed to reclaim its memory, unless there is garbage collection available like in Java Card Connected. Garbage collection may not always be supported and in any case the run-time creation and deletion of objects could lead to memory problems and errors that may not have been reproduced during applet testing. Pragmatically, most applications will have to create objects and so this should be done once only, during the applet installation phase. Furthermore, if data buffers are created, these should be re-used as much as possible (and therefore given generic names). Comments of course do not consume any smart card memory at all, and therefore should be used as much as possible, especially given the lack of object-orientation.

Java Card Classic and Connected are very strict on types (short, byte, int etc.) and generally this is a good thing, however, the fact that in Java Card Classic bytes are signed and not unsigned will give the novice developer no end of trouble. To get around the sign issue, source code may contain a surprising amount of casting and byte masking and sometimes a developer will have added utility functions to take care of unsigned byte manipulation.

On the plus side, exception handling is a blessing for Java Card development as it allows efficient data checking without exhausting too much memory. For example, if the data supplied by the terminal is not correct (see the class and instruction tests in Listing 10.2) then simply throw an exception. This will automatically generate the status message response with the SW1/SW2 status of your choice. There is no need to extract yourself from nested function calls, as the Java Card platform will automatically tidy up the stack, etc.

So far we have talked about the generic Java Card but as we wish to focus on the telecommunications version we need to consider Java SIM.

10.6 Java SIM

Java SIM is simply a Java Card with pre-installed SIM applet and SIM-related APIs providing the same functionally as native GSM SIM or 3G USIM, governed by the 3GPP (originally ETSI) specifications [8, 9]. On Java SIM each SIM Toolkit application exists as a separate Java Card applet.

GSM operators fought hard to get Java SIM for a number of reasons:

1. To make their dual/multiple smart card sourcing easier (e.g. develop once, deploy onto any card).
2. To encourage an open platform with lower cost products and remove barriers for new entrants to the market.
3. To create third-party development opportunities.

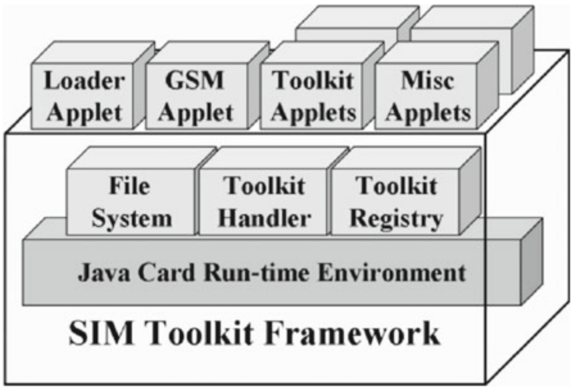


Fig. 10.3 Java SIM architecture

4. To leverage from Java security and management functionality that had achieved high accreditation ratings in other business areas (banking).

The Java SIM Framework is shown in context within Fig. 10.3. In practice the additional functionality is accessed via two extra packages as defined in the standards [10]: `sim.toolkit` and `sim.access`.

10.6.1 Sim.toolkit

This is the core package that manages SIM Toolkit-related activity, and provides a number of classes including;

- ProactiveHandler,
- ProactiveResponseHandler,
- ToolkitRegistry,
- EnvelopeHandler,
- EditHandler &
- ViewHandler.

The first two classes have “Proactive” in their names and this is quite significant. SIMs can be used as slave devices simply satisfying requests from the network or mobile terminal but the proactive SIM feature gives them the opportunity to take temporary control. This is a necessary capability for SIM-based applications to drive a sequence of actions although they still need some event(s) to trigger them to run. Events are processed by a SIM Toolkit handler process, e.g. `public void processToolkit(byte event).`

Rather like when `process(APDU)` is triggered by a general incoming APDU, `processToolkit()` is triggered by notification of an “event”. This can be seen in the code segment of Listing 10.3, line 19.

```
package com.o2.JavaCard;

import javacard.framework.*;
import sim.toolkit.*;
import sim.access.*;

public class MyFirstJavaCardApplet extends Applet {

    private byte[] text = {(byte)'H', (byte)'e', (byte)'l', (byte)'l
        ', (byte)'o'};

    private MyFirstJavaCardApplet() {
      register();
    }

    public static void install(byte bArray[], short bOffset, byte
        bLength){
        new MyFirstJavaCardApplet();
    }

    public void processToolkit(APDU apduHandle){
      if(event == EVENT_PROFILE_DOWNLOAD){
          ProactiveHandler ph = ProactiveHandler.getTheHandler();
             ph.initDisplayText((byte)0, DCS_8_BIT_DATA, text, (
                 short)0, (short)text.length);
             ph.send();
        }
    }
}
```

Listing 10.3 Example SIM Toolkit Applet

The example is only interested in the PROFILE_DOWNLOAD event which informs the SIM of the mobile terminal capabilities. This event occurs only once and early on during initialisation. When the event is triggered, the applet simply places an unoriginal text message "(Hello)" onto the handset display. There are quite a number of SIMToolkit events as shown in Table 10.1 and more fully described in the standards [10].

As can be seen in Fig. 10.3 there can be multiple SIM Toolkit applets and so when one of the events in Table 10.1 occurs, it has to be routed to the correct handler. The SIM Toolkit Framework takes care of this by using the Toolkit Registry. When a Toolkit applet is installed it not only registers itself to the JCRE but also to the Toolkit Framework, creating a Toolkit registry entry. The entry indicates the events that should be routed to the particular applet. Most commonly, a Toolkit applet would register for the appropriate set of events only once but it is possible to change the set dynamically.

Whatever the purpose of the SIM Toolkit applet and the event(s) that it handles, it is likely that the applet will need to access some of the files stored in the SIM and this requires an additional package.

Table 10.1 SIM Toolkit events

Received SMS handling	FORMATTED_SMS_PP_ENV
	FORMATTED_SMS_PP_UPD
	UNFORMATTED_SMS_PP_ENV
	UNFORMATTED_SMS_PP_UPD
	FORMATTED_SMS_CB
	UNFORMATTED_SMS_CB
Menu selection handling	MENU_SELECTION
	MENU_SELECTION_HELP_REQUEST
Call and SMS send control	CALL_CONTROL
	SMS_MO_CONTROL
Utility	UNRECOGNISED_ENVELOPE
	STATUS_COMMAND
	TIMER_EXPIRATION
Mobile process events	EVENT_DOWNLOAD
	MT_CALL
	CALL_CONNECTED
	CALL_DISCONNECTED
	LOCATION_STATUS
	USER_ACTIVITY
	IDLE_SCREEN_AVAILABLE
	LANGUAGE_SELECTION
	BROWSER_TERMINATION
	CARD_READER_STATUS
Determining terminal capabilities	PROFILE_DOWNLOAD

10.6.2 Sim.access

This package provides access to the GSM SIM file system on the card without compromising data integrity (i.e. an application accessing data at the same time as a terminal APDU command). It includes the SIMSystem and SIMView classes and provides methods for reading and writing to individual files, referenced by file identifier (FID) constants. The methods available in the SIMView class are listed in Table 10.2 and one can see how they mirror the commands defined in GSM 11.11 [8].

To use the methods provided by the SIMView to access the telecoms files, requires that a toolkit applet can obtain an overall view of the GSM File system. As the files are not stored within the toolkit applet it is first necessary to get a handle to the GSM file system. This is achieved by using the single method of the SIMSystem class; `getThe SIMView()`. A simple example of using SIMSystem and SIMView is shown in Listing 10.4.

Table 10.2 SIMView methods matching GSM 11.11 commands

short increase(**byte**[] incr, **short** incrOffset, **byte**[] resp, **short** respOffset)
void invalidate()
short readBinary(**short** fileOffset, **byte**[] resp, **short** respOffset, **short** respLength)
short readRecord(**short** recNumber, **byte** mode, **short** recOffset, **byte**[] resp, **short** respOffset, **short** respLength)
void rehabilitate()
short seek(**byte** mode, **byte**[] patt, **short** pattOffset, **short** pattLength)
void select(**short** fid)
short select(**short** fid, **byte**[] fci, **short** fciOffset, **short** fciLength)
short status(**byte**[] fci, **short** fciOffset, **short** fciLength)
void updateBinary(**short** fileOffset, **byte**[] data, **short** dataOffset, **short** dataLength)
void updateRecord(**short** recNumber, **byte** mode, **short** recOffset, **byte**[] data, **short** dataOffset, **short** dataLength)

```
    ...
    GsmHandle = SIMSystem.getTheSIMView();
    ...
    GsmHandel.select((short)SIMView.FID_DF_GSM);
```

Listing 10.4 SIMSystem and SIMView Class Usage Example

Note there are several reasons why access to a file may be prevented or result in an error or failure, so there is also a `SIMViewException` class to handle these situations. In fact when developing SIM card code in general, there are a lot of opportunities to introduce build-time and run-time errors. Because the platform does not have its own user interface to provide step-by-step diagnostics from a running applet it is hard to diagnose when something is going wrong. This is especially difficult for SIMs because there are so many variations of handsets with different SIM interface capabilities that are not always bug free. Bearing in mind that a new SIM card may be deployed in its millions, it is vitally important that we have powerful tools to not only develop but also test and analyse card applications.

10.7 Application Development Tools

There are a number of dedicated tools on the market for those wishing to develop Java Card and Java SIM applets. These tools break down into the following categories:

- Compilers and Integrated Development Environments
- Simulators
- Protocol Analysers (Spy Tools)
- Utilities

10.7.1 Compilers and Integrated Development Environments

The starting point for any software development is the compiler and Integrated Development Environment (IDE). These tools allow you to create the source code for your application, with eventual compilation down to executable code for loading onto a smart card. One of the advantages of Java is that even though we are concerned primarily with the Java Card variant we can use nearly all of the mainstream Java IDE products on the market, together with a dedicated Java Card plug in (to support the particular semantics of Java Card). Two of the more popular and easily obtained IDEs are Eclipse [11] and NetBeans [12]. Both of these tools allow you to download and install plug-ins, to support Java Card development. NetBeans natively supports Java Card 3 with a easy to use simulation environment. Whereas, Eclipse provides a customisable environment that a developer can mould to his/her requirements.

There are also some very powerful and specialist tools from the Java SIM Card Vendors like Giesecke and Devrient and Gemalto. The tools from Java SIM Card Vendors are not necessarily cheap to buy and are aimed at the serious smart card developer, which is often the customer for the cards, i.e. the network operator.

10.7.2 Simulators

Unlike other forms of software creation, the development and testing cycles on smart card platforms can be extremely tedious as the executable platform is not the same as the development platform. For example, the executable code (in this case Java bytecode, encapsulated in a CAP file) has to be downloaded onto a physical smart card placed in a reader and then the card is removed and placed into a terminal device (e.g. a GSM mobile phone) in order for the executable code to run. This process involves a high degree of physical card manipulation and potentially lengthy download times (in the order of a minute or two), greatly inhibiting the typical development flow of repeated code tests and modifications. It can also lead to a temptation to reduce testing, which raises the spectre of millions of faulty cards in customer hands. Fortunately the majority of IDEs come with some level of simulation and debug facility. Simulators are a real boon for Java Card development in that a software model of a Java Card (or Java SIM) allows near-instant download, installation and execution of Java Card applets. You still need to exercise the applet in some way and even the most basic simulators include or provide access to an APDU Scripter. The scripter allows systematic generation and issuing of APDU commands to the smart card and will record the responses. The more advanced scripter tools will react to the responses so that the next script depends on the result of a previous response. A Scripter tool gives very precise control over the commands and data sent to the card. However, it is not well suited to the faithful generation of the many complex command and response sequences that would occur in real life. For example when a mobile phone powers up there are many hundreds of messages to and from the SIM card. In order to cope with this problem and make the simulations as realistic as possible, the more elaborate SIM card simulator tools also include a mobile phone simulator. Usually the phone appears as a realistic looking graphic on the computer screen allowing the computer operator to "press" the virtual buttons on the keypad and generally use the phone as if it was real. Exercising the normal phone functionality generates all the associated message exchanges between the phone and the SIM card. The simulator would normally display and log these exchanges and offer the means to breakpoint the simulation at some point of investigation. A further enhancement is to simulate network and applications servers so that an entire system solution may be tested and debugged all from a simple and convenient computer interface.

Whilst simulators are tremendously useful, some words of caution are necessary. Simulation can speed up early development and test cycles, but it is definitely not a substitute for testing real SIMs in real phones. A particular SIM is likely to be a well-standardised device, however its behaviour is rarely identical to its virtual model. For example a CAP file that loads and runs nicely on the virtual card might be rejected by the real card loader tool or perhaps if loaded/installed, runs differently because of some differing assumptions about variable types. Perhaps a starker example is that you could load and script test a SIM-like card application on a virtual T = 1 protocol card, whereas it will never work in a GSM phone as the T = 0 protocol is required. However, the big problem is not so much with the SIM, but with the mobile phones. Their SIM messaging sequences are surprisingly diverse and cannot be confidently modelled as they depend on make, model, capability, usage and software release. The great challenge when testing a new SIM application is to ensure it will work with the thousands of phones used on the network in all possible sequence combinations, e.g. authenticating, making calls, receiving SMS, using a menu, etc. If simulation cannot replace device testing it can ease the transition from the virtual to real world by some flexible options. For example, you might test your virtual model with scripts then load a real card and use the same scripts. Alternatively if your simulated card works with a simulated phone, you could then move to a real card and exercise it from the simulated phone. Despite all this good work there will come a time when the simulations work fine but the real card in the real phone does not. Then it is time to reach for a different kind of tool.

10.7.3 Protocol Analysis (Spy) Tools

There is a problem when trying to diagnose a problem that involves mobile phone and SIM interaction. The SIM has no user interface to help you and although the mobile could display something, it has little or no debug and diagnostic facilities to tell you when things go wrong. Mobile phones also behave very differently when they have trouble communicating with the SIM. Some lock-up completely, some report invalid SIM present, others continue on as if nothing untoward has happened. Fortunately, we have an accessible physical interface between the card and terminal, so it is possible to use a protocol analyser to effectively "spy" on all commands and associated responses. This is possible given the physical interface between the mobile phone and SIM, which can be electrically probed without affecting communication. The tools tend to use a simple desktop computer for user-interface functions with the specialist real time measurement and logging hardware in an external unit. These tools come with a range of adaptor leads that can plug into the SIM socket of a mobile phone and lead the contacts to the measurement unit. The SIM under test is then inserted into the external unit rather than the phone. Thereafter the phone works as if the SIM was directly installed and all message and responses can be logged, displayed and analysed. There are several companies providing such spy tools, including Integri [13].

It should be emphasised that whilst these products are commonly referred to as spy tools their usage is quite legitimate, being aimed at commercial development and testing. Attackers could of course use the tools and so developers should ensure that their application security is not compromised by unauthorised access to the SIM/mobile interface.

10.7.4 Utilities

Aside from sophisticated IDEs, simulators and spy tools there are lots of standalone utility programs that a developer may find useful. Good utilities are fast, easy to use and very importantly are card vendor independent. Some typical utilities are listed in Table 10.3.

Table 10.3 Utility tools

Utility	Description
APDU scripter	This allows the user to create custom APDUs, send them to the smart card and log the responses. This capability is often included in the main IDE, however the standalone versions can be less restrictive, e.g. when controlling the class and instruction bytes
File system viewer	This is equivalent to a kind of Windows Explorer for the smart card file system. Depending on security permissions the user can normally see the overall directory structures, file contents and file headers
Profile editor	This is an extension to the File System Viewer and lets the user modify some files—when permitted by the security settings
Profile checker	The configuration and contents of a SIM are known as a profile. Profiles can become very complex, making them difficult to compare. A profile tool can capture and compare profiles from different cards providing the user with a clear report on the differences
Applet loader	A Java applet is loaded onto a smart card in the form of a CAP file, which is produced by the IDE. The loader handles the message exchange and security associated with loading the applet onto the card. An Applet Loader is often included with the IDE but there are also standalone versions

10.8 Mobile Phone Applications and the SIM

A modern mobile phone is a powerful multimedia computer platform with a wireless connectivity link for exchanging calls, messages and general Internet access. The functionality of these platforms is no longer fixed and it is possible to download new applications to support value added services. The technology is not as well standardised as in the smart card world, as aside from the far greater complexity of mobile phones compared to smart cards, user choice is greatly influenced by the functionality and style of a phone and so it is clearly a differentiator between manufacturers. Because of the commercial pressures and differences of opinion over what is the "best" platform there are number of variants in the market. The existence of these variants is rather a problem for application development and the associated widespread introduction of new value added services. Fortunately, the Open Mobile Terminal Platform (OMTP) [14] organisation is trying to address these issues. Discussing the relative merits of say Nokia/Symbian, Windows mobile or any other flavour is way beyond the scope of this chapter so we will stick with the Java approach to continue our description.

Some mobile phones support a Java environment called the Java Mobile Edition. It was once referred to as J2ME (Java 2 Platform Micro Edition). Whereas our applications on the Java SIM were called applets the phone-based applications are called MIDlets. A good question is why would a mobile phone with all its processing, interface and communications capability want to interact with a lowly SIM? Well the answer relates mainly to security related functionality, but also to control, configuration and management.

Mobile phones do not have a great reputation when it comes to security. The most common example is how GSM handset network locks can be removed for a small amount of money, or perhaps the theft of personal information via Bluetooth links. There may be little physical security to stop the better equipped attacker from examining and modifying both data and circuitry. Therefore, it would be useful to exploit the services of a hardware security module that has been designed to resist such attacks. As there is a SIM designed for just such a purpose it is not surprising that a MIDlet API has been designed that can make use of it.

10.8.1 SATSA

SATSA stands for Security And Trust Services API. It is sometimes referred to as JSR177 [15] after the name of the expert group that defined it. It is not aimed solely at exploiting a SIM, but a security element (SE) in general. The idea is that the SE will help provide secure data storage, some standard cryptographic primitives and an execution environment for custom security functionality. The API is split across four packages as shown in Table 10.4.

Table 10.4 SATSA API packages

Package name	Description
SATSA-APDU	This allows the MIDlet to communicate with the SIM. As it is at the APDU level interface, it offers a great deal of control and flexibility
SATSA-JCRMI	This is Java Card Remote Method Invocation—basically it means that the MIDlet can remotely run a program (method) implemented within a Java SIM applet
SATSA-PKI	This is intended for digital signing as part of a Public Key Infrastructure system
SATSA-CRYPTO	This is to provide a toolbox of useful cryptographic utilities; encryption/decryption, verification, hashes etc.

A complete explanation of the API packages is beyond the scope of this chapter and so our treatment will be restricted to the main features of SATSA-APDU and SATSA-JCRMI, which fit well with our previous discussions on Java SIM.

10.8.1.1 SATSA-APDU

To communicate with the SIM, the MIDlet must establish a logical connection with it. This is done by using "APDUConnection()" and specifying the appropriate Application ID (AID). In fact if the SIM supports it, the MIDlet can establish multiple logical channels. The standards also allow for the case of the MIDlet having access to multiple SEs and communications over the logical connection follows the conventional ISO7816-4 [3] standards. The main component of SATSA-APDU is the "javax.microedition.apdu" package that includes the "APDUConnection" interface, (additionally SATSA-APDU supports some exception classes).

The "APDUConnection" interface contains a number of methods as shown in Table 10.5.

The "exchangeAPDU()" method is of most interest and there are couple of points to note. First, the class byte field (CLA) in the APDU contains the logical channel number rather than some industry specific value. Second, this method will block until the card responds and so an operational timeout/interrupt is need to stop the process hanging. An APDU level interface provides a lot of flexibility, however, it does mean that the MIDlet (and the developer implementing it) has to know quite a lot about smart cards and protocols. A higher level interaction at the applet/method level is perhaps more appropriate and more in keeping with the Java principle of being agnostic to the lower layers of implementation. Remote method invocation is a way to achieve this.

Table 10.5 APDU connection methods

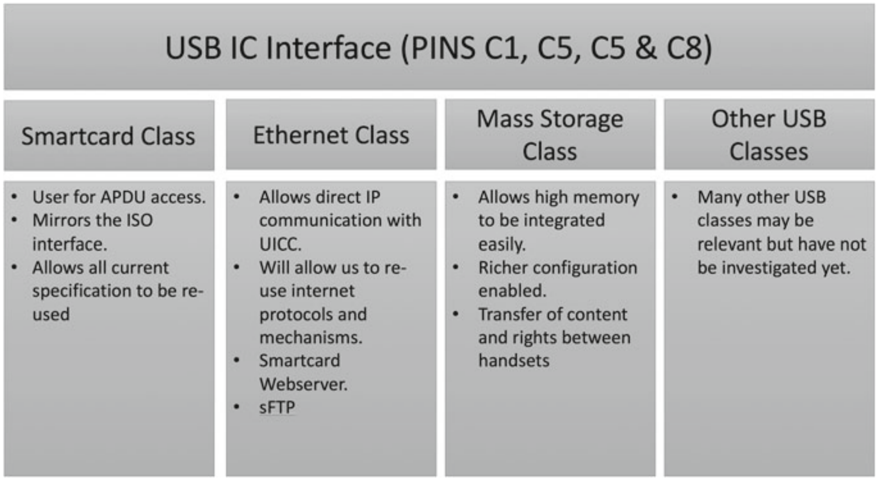
Method	Description
Public byte[] changePin(int PinID)	Combined prompt/process for the user to enter a new PIN and update on SE
Public byte[] enablePin(int PinID)	Combined prompt/process for the user to enable a particular PIN and update on SE
Public byte[] disablePin(int PinID)	Combined prompt/process for the user to disable a particular PIN and update on SE
Public byte[] unblockPin(int blockedPinID, int unblockingPinID)	Combined prompt/process for the user to unblock a particular PIN
Public byte[] enterPin(int PinID)	Combined prompt to the user to enter a PIN and its subsequent verification by the SE
Public byte[] exchangeAPDU(byte[])	Main method for APDU exchange
Public byte[] getATR()	Provides the card Answer To Reset information supplied by the SE when it was last reset (also acts as a card present check)

10.8.1.2 SATSA-JCRMI

SATSA-JCRMI provides an API to invoke a remote application implemented on the SIM card. Aside from exception support and some general RMI functionality the main package is “javax.microedition.jcrmi”. The package has two interfaces that are “Java CardRMIConnection” and “RemoteRef”. The first interface is rather similar to the “APDUConnection” in that it establishes a logical channel for communicating with an application on the SIM card, identified by its AID. There are also are similar range of PIN management functions. The method “getInitialReference()” is used to get an initial reference to the SIM application so its methods may be used. The “RemoteRef()” interface represents a handle or reference to a remote object, whose most notable method is “invoke()”, which as the name suggests is just what we were looking for; a means to invoke a remote method of an object stored within a smart card applet.

10.8.2 More Recent Development

In last couple of years, especially after the advent of smart phones, the overall ecosystem of the mobile industry has seen numerous technological improvements. Beside technology, the uptake of smart phones with highspeed interconnectivity is moving many services that were available on traditional devices like Personal Computers (PCs) and smart cards to mobile phone. Furthermore, SIM is being deployed in many different domains/devices—beside mobile phones—like in PCs for two-factor authentication, smart meters and Internet-of-Things (IoT).



USB IC Interface (PINS C1, C5, C5 & C8)

Smartcard Class	Ethernet Class	Mass Storage Class	Other USB Classes
• User for APDU access. • Mirrors the ISO interface. • Allows all current specification to be re-used	• Allows direct IP communication with UICC. • Will allow us to re-use internet protocols and mechanisms. • Smartcard Webserver. • sFTP	• Allows high memory to be integrated easily. • Richer configuration enabled. • Transfer of content and rights between handsets	• Many other USB classes may be relevant but have not be investigated yet.

Fig. 10.4 USB IC interface

In addition, some of the salient new UICC features listed in the ETSI TS 102.412 are listed and discussed below:

High Speed Interface:

One of the limitations of traditional smart cards is their communication interface. With increase in the performance of other devices and potential requirement for integration into more powerful devices—smartphones and IoTs etc., much faster interface speeds (and source clock) is crucial. This is being addressed in ETSI TS 102.412, that has proposed a high speed interface. Potentially this might add 8 Mb of USB channel to replace the ISO interface. In addition, this new high speed interface might also support the IP layer protocols. Figure 10.4 illustrates the USB interface features and capabilities.

Contactless Interface:

Near Field Communication (NFC) was jointly developed by Philips (NXP) and Sony for contactless communication. Later in 2004, Nokia, Philips and Sony founded the Near Field Communication Forum that maintains the NFC specifications.

NFC is a low frequency radio wave that operates at 13.56-MHz spectrum and has an operational distance of less than 10 cm. This requires that the NFC devices have to be very close to each other in order to communicate. There can be two basic types of NFC devices. The first type generates the low frequency radio wave field and for this, they should have an internal power supply; this is known as an active NFC device. The second type is a device that generates power from the magnetic inductive coupling when it comes close to the NFC radio field; such devices are termed as passive NFC devices. As these devices are dependent on the active devices for the power supply, they do not have any internal power source of their own. An NFC device can act as both active and passive device as required, and this ability gives NFC-based devices a unique capability as compared to other contactless communication technologies. In addition, NFC devices can operate in three different modes as discussed below:

Reader/Writer Mode

In this mode, an NFC device can read and write data to a NFC passive device. Examples could be advertisement tags that can be attached to a particular advertisement board/message on railway stations, tube stations, airport and theatre. If users require further information regarding the advertisement, they would just require to touch their NFC device to the tag, and it would transmit the stored information. That can be a web address (Uniform Resource Locator—URL), date and time of the event, purchase information or anything else that relates to the advertisement.

Peer-to-Peer

This is the default NFC communication mode, in which two NFC devices establish a bi-directional connection to exchange data [16, 17]. It removes the cumbersome process of connecting devices with each other on protocols like Bluetooth, and Wireless networks as depicted in Fig. 10.5 taken from NFC forum specifications. In this mode, there would be one server NFC device (so-called NFC peer-to-peer target) that holds the data (details on how to establish the connection using the desired protocol) and the other devices are termed as client NFC devices (so-called NFC peer-to-peer

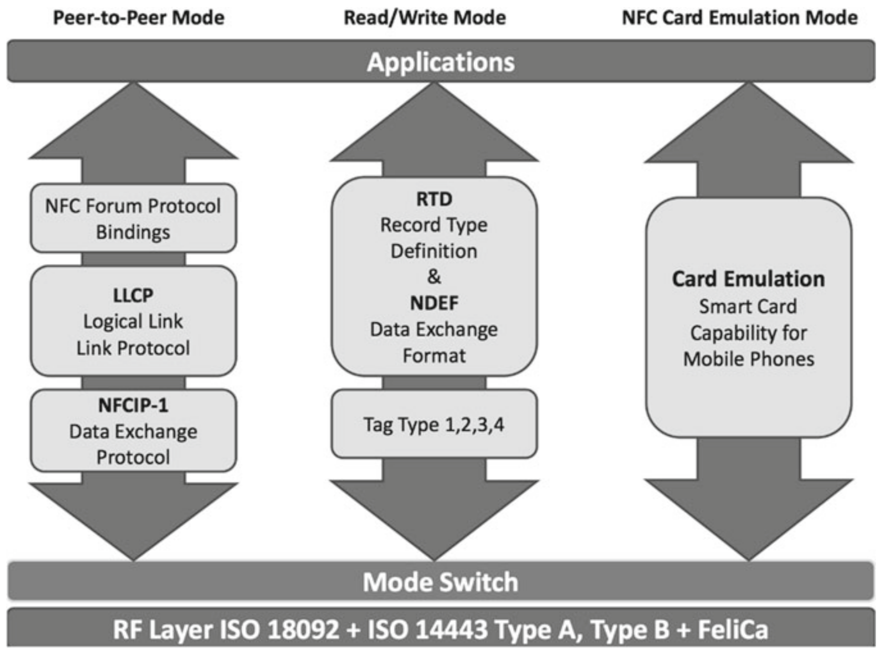

Peer-to-Peer Mode
Read/Write Mode
NFC Card Emulation Mode
Applications
NFC Forum Protocol Bindings
LLCP Logical Link Link Protocol
NFCIP-1 Data Exchange Protocol
RTD Record Type Definition & NDEF Data Exchange Format
Tag Type 1,2,3,4
Card Emulation Smart Card Capability for Mobile Phones
Mode Switch
RF Layer ISO 18092 + ISO 14443 Type A, Type B + FeliCa

Fig. 10.5 NFC Forum technology architecture

initiator). When the client NFC device comes in the server NFC device's range, they establish a connection and transfer necessary setup details using NFC Data Exchange Format (NDEF) for pairing two devices.

Card Emulation:

In this mode, NFC devices act as a contactless smart card or card reader in compliance with the ISO 14443 [7]. In the case of the card emulation mode, an external reader is unable to distinguish between a smart card and an NFC device. This mode is useful as NFC device could be utilised in any service sector where traditionally smart cards are already deployed (i.e. banking, transport and building access, etc.). During the course of this chapter when we refer to NFC, we actually mean the card emulation mode of the NFC. In this scenario, a mobile phone has a built in NFC antenna that a Secure Element (SE) in the mobile phone can use to emulate the contactless smart card interface. The applications that interact with the external readers are stored in the Secure Element and from the reader's point-of-view, they are on contactless smart cards.

Secure UICC—Terminal Interface:

Secures the terminal to UICC interface so secure transactions can take place such as Digital Rights Management (DRM). We should keep in mind that this is not a completely new service but an improvement of an existing service that included strengthened security. By this improvement the interface is opened up to other services—providing a flexible framework for non-Mobile Network Operator services to interact with the SIM.

Confidential Applications:

This allows the creation of firewalled third party areas. These third-party areas can be used and managed safely by third parties. The UICC provides the service to third parties to securing manage their services (similar to the GlobalPlatform specification for domain management). As stated by the ETSI TS 102 226 (v9.4.0) the confidential application management includes the follow services, depending upon the scenario:

- Application Provider Security Domains (APSDs) provides cryptographic services for applications and their provides. This cryptographic services include secure channel protocol for confidential messaging.
- The APSDs should provide the Over-the-Air (OTA) capabilities to individual applications and application providers can communicate with each other independently.
- The APSDs should also support in required the GlobalPlatform secure channel capabilities without having OTA capabilities themselves.
- Require a Controlling Authority Security Domain (CASD) that manages the security services involving the generation of a security domain for third parties.

Smart Card Web Server

The Smart Card Web Server (SCWS) specifies how the SIM card can act as a WAP server. The main advantages of this are shown below:

- Uses built-in browser of mobile device to render data
- For simple web server content quite high interoperability
- OTA provision of user interface

Some disadvantages of the SCWS include:

- New technology in UICC
- Difficult to define the exact look and feel of the interface
- Different versions of web pages for different browsers and screen
- Not widely supported by phones

10.9 The Smart Phone Era

As discussed before, the advent of smart phones has—with no exaggeration—revolutionised not only the mobile handsets but also the way different services are utilised by individual users. As shown in Fig. 10.6, smart phones have a number of new and existing features including Host Card Emulation (HCE), embedded UICC (eUICC), Near Field Communication (NFC) and Trusted Execution Environment (TEE). These elements are not discussed in this chapter in detail, as they are comprehensively described in other parts of this book.

With the introduction of a large number of services, it is important to establish and manage trust in the smart phone's environment. Some core elements of trust on smart phones are discussed below:

1. Hardware Security: dedicated silicon elements present on a smart phone enable this. These elements include UICC, eUICC and embedded Secure Element

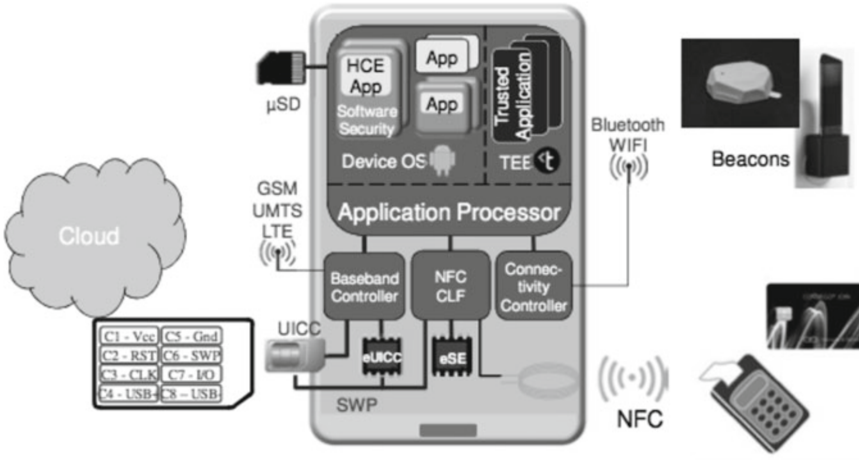


Fig. 10.6 Modern smart phone (generic) architecture

(eSE). These elements provide secure storage for cryptographic keys, and can securely execute applications include cryptographic algorithm and privacy services (encrypted radio interface to protect user's telephone calls).
2. Trusted Execution Environment: a special hardware component that allows selected application to execute with strong isolation, enforced by hardware measures. Trusted Execution Environment (TEE) and HCE are discussed in detail in Chap. 18.
3. Host Card Emulation (HCE): A communication framework developed for Google Android devices that allows applications in software to communicate directly with the NFC chip on the smart phone. This enables a mobile application to emulate a payment application on a smart phone.
4. Software Security: This enables the software to build an efficient protection mechanism in software—without relying on the security services provided by any underlying hardware. Examples includes Code Obfuscation, White Box Cryptography and Device Fingerprinting.

10.10 Concluding Remarks

This chapter has attempted to provide the reader with a brief overview of a wide range of issues associated with smart card application development and how fast smart phone technology is changing is having an effect on the embedded development processes. The focus has been restricted to Java-based technology and the reader is referred to Chaps. 1 and 4 for alternative technologies and development methods.

The Java SIM has been used as the primary platform, partly because SIMs have a superset of Java Card capabilities, but also because a lot of Java SIMs have been deployed in the market. We have seen that smart cards have historically been treated as very restricted application platforms, although support for Java is bringing them closer to mainstream development. In fact, the tools now available to developers are very sophisticated and advanced SIM Toolkit features such as event and file handling can be exploited by SIM applications via the Java Card Framework, Unfortunately SIM-based applications have never fully realised their potential partly due to problems with mobile handset support of critical standards. However, the most interesting service ideas see the handset and SIM working together as parts of a system solution, underpinned by the SIMs security qualities. Within Java mobiles supporting SATSA the problems of co-operating with the SIM have already been addressed and standardised in the form of APIs, which permit application level and APDU message level interaction. Taking a common approach to development by using Java in both the mobile handset and also the SIM makes life easier for developers and perhaps as the SIM evolves towards a less restricted and IP-based platform, the boundaries between phone and SIM development may disappear.

It is clear that today's standards already contain very powerful features for both SIM, eUICC, eSE and mobile Phone application development. There are more than enough sophisticated tools to exploit these features and so the challenge is to get consistency across the various types of mobile phone platform and the SIM interface features that they support. This is well worth the effort as it could enable a broad range of value added mobile services underpinned by the hardware security module characteristics of the SIM. In fact it looks as if the next decade is going to be an interesting time for smart cards in general, as their world is opened up to more mainstream development and their capabilities, including processing power and memory increase rapidly. With all this change, increased openness and added technical complexity, it is important that the inherent smart card characteristics remain. Security, reliability and controllability will be even more essential to future system solutions.

References

1. Oracle Incorporation, *Java Card Platform Specification v 2.2.2,* More Information Available via http://www.oracle.com/technetwork/java/embedded/javacard/downloads/index.html, Cited 22 Aug 2016.
2. ETSI, SIM Technology, More Information Available via http://www.etsi.org/WebSite/Technologies/SIM.aspx, Cited 22 Aug 2016.
3. ISO, ISO7816-X, *Identification cards,* More Information Available via http://www.iso.org/, Cited 19 Aug 2016.
4. R.N. Akram, K. Markantonakis and K. Mayes, An Introduction to Java Card Programming, in Secure Smart Embedded Devices, Platforms and Applications, eds: K.Markantonakis and K. Mayes, Springer (2014).
5. Java Card Platform Specification; Application Programming Interface, Runtime Environment Specification, Virtual Machine Specification. Online (2015). http://www.oracle.com/technetwork/java/embedded/javacard/overview/default-1969996.html Cited 22 Aug 2016.
6. NSA Suite B Cryptography. Online. http://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml Cited 22 Aug 2016.
7. ISO/IEC 14443: Identification Cards—Contactless Integrated Circuit(s) Cards - Proximity Cards, Part1: Physical Characteristics, Part 2: Radio Frequency Power and Signal Interface, Part3: Initialization and Anticollision, Part 4: Transmission Protocol (2008). http://www.iso.org/iso/catalogue_detail.htm?csnumber=39693 Cited 22 Aug 2016.
8. 3GPP, *Specification of the SIM-ME Interface Rel.99 3GPP TS 11.11* V8.14.0 (2007-06), More Information Available via http://www.3gpp.org/, Cited 22 Aug 2016.
9. 3GPP. *Specification for the SIM Application Toolkit for the SIM-ME Interface Rel.99, 3GPP TS 11.14* V8.18.0 (2007-06), More Information Available via http://www.3gpp.org/, Cited 22 Aug 2016.
10. 3GPP, *SIM API for Java Card Stage 2 Rel.99 3GPP TS 03.19* V8.5.0 (2002-09), More Information Available via http://www.3gpp.org/, Cited 22 Aug 2016.
11. Eclipse, Open Development Platform, More Information Available via http://www.eclipse.org/, Cited 19 Aug 2016.
12. NetBeans, IDE 5.5.1, More Information Available via http://www.netbeans.org/, Cited 19 Oct 2016.
13. Integri, More Information Available via http://www.integri.com/, Cited 19 Aug 2016.
14. Open Mobile Terminal Platform (OMTP), More Information Available via http://www.omtp.org/, Cited 22 Aug 2016.
15. JSR177 Experts Group, *Security and Trust Services API (SATSA) v1.0.1 for J2ME,* More Information Available via https://jcp.org/aboutJava/communityprocess/final/jsr177/index.html, Cited 22 Aug 2016.

16. ECMA-373: Near Field Communication Wired Interface, European Association for Standardizing Information and Communication Systems (ECMA), Online (2006). http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-373.pdf Cited 22 Aug 2016.
17. ISO/IEC 28361: Near Field Communication Wired Interface (NFC-WI), International Organization for Standardization (ISO), Online (2008). http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=44659&ics1=35&ics2=100&ics3=10 Cited 22 Aug 2016.
18. 3GPP, *Characteristics of the USIM Application Rel.7 3GPP TS 31.102* V7.10.0 (2007-09), More Information Available via http://www.3gpp.org/, Cited 22 Aug 2016.
19. Comprion, More Information Available via http://www.comprion.com/, Cited 22 Aug 2016.
20. ETSI SCP Group, *SCP Specifications,* More Information Available via http://portal.etsi.org/docbox/scp/scp/Specifications/, Cited 19 Aug 2016.
21. GSM & UMTS—*The Creation of Global Mobile Communication*—Wiley 2002.

Chapter 11
OTA and Secure SIM Lifecycle Management

Joos Cadonau, Danushka Jayasinghe and Sheila Cobourne

Abstract In the GSM mobile communication industry, the end-user is referred to as the subscriber and identified in the operator's network using the Subscriber Identity Module (SIM). In the third-generation network 3G, the equivalent application is called Universal Subscriber Identity Module (USIM) card; although by convention we use SIM for both, unless a distinction is needed. A SIM card is a removable smart card for mobile phones. A mobile network operator is a telephone company that provides services for mobile phone subscribers. The SIM card is a managed platform, belonging to the operator's network. It offers to store operator specific but also subscriber-related data. SIM cards are in use for a long time, compared to the handset and other entities in the network. Therefore, SIM card data—operator or subscriber dependent—changes over time and needs Over-the-Air (OTA) management. Customers cannot be asked to visit an operator shop for data management; Over-the-Air updates using the SMS as a bearer are the only possibility for mass updates. This implies certain security requirements which are specified in the 3GPP/ETSI standards. Also, the current bandwidth offered by SMS limits the range of possible adaptations and requires the mobile network operator to have a flexible Over-the-Air system, adapted to their needs. The Over-the-Air management is only one stage of the SIM life cycle. To be able to launch new services during the life cycle of a SIM card, the whole SIM life cycle has to be planned carefully. There exist systems that support the operator in knowing in real time the status of a SIM card in all phases of the SIM life cycle.

Keywords Over-the-Air OTA · SIM Application Toolkit SAT/STK · SIM Lifecycle Management · SIM/USIM Card · OTA Performance Limitations · Public Land Mobile Network PLMN

J. Cadonau
iQuest Schweiz AG, Frankfurt am Main, Germany
e-mail: joos.cadonau@iquestgroup.com

D. Jayasinghe (✉) · S. Cobourne
Smart Card Centre, Information Security Group,
Royal Holloway, University of London, Egham, UK
e-mail: Danushka.Jayasinghe.2012@live.rhul.ac.uk

S. Cobourne
e-mail: Sheila.Cobourne.2008@live.rhul.ac.uk

K. Mayes and K. Markantonakis (eds.), *Smart Cards, Tokens, Security and Applications*, DOI 10.1007/978-3-319-50500-8_11

11.1 Introduction

The (Universal) Subscriber Identity Module (SIM) card's properties as a tamper-proof device and as a unified storage facility for secure information, as well as its ability to access functions in the mobile handset and the network, make it a key component. The SIM is a part of the operator's network, and as such, it holds vital information that must be managed and provisioned. Market requirements have introduced solutions that ensure that each and every card in circulation has the optimal configuration and set of services embedded at all times. The SIM is the only linkage point between the network operator and the subscriber, thus making it imperative for the operator to have control over which networks their subscribers roam onto and to manage the subscriber's access to Value Added Services (VAS) via SIM-based menus such as Dynamic SIM Toolkit (DSTK).

The SIM card not only holds operator-specific data, but also stores subscriber-related data such as the subscriber's phone book, the last numbers dialled or text messages. Subscriber-based data is not static, but needs to be updated or saved regularly. This requires Over-the-Air (OTA) management, as well as building additional services around the subscriber-related data such as a phonebook backup functionality.

As SIM cards continue to evolve with more memory capacity and functionality, they will continue to be an important part of an operator's network. Over-The-Air becomes essential in managing data on the SIM such as Java applets, Public Land Mobile Network (PLMN) roaming lists, dynamic SIM Toolkit (STK) menus and the subscribers' personal information.

For the subscriber, SIM is a personal, portable gateway to new services; for the operator, it is the key to security and differentiation in the world of 3G mobile services.

11.2 The SIM Card as a Managed Platform

The SIM card is a key element in the operator's network to identify the subscriber and to authenticate the SIM in the network. There are a number of security features provided by the SIM card to guarantee those processes and to cipher the information on the radio path. The SIM card can therefore be seen as a secure element in the operator's network which also makes it attractive to hold additional services requiring security such as banking applications.

11.2.1 Common Stored and Managed Data

Table 11.1 gives an overview (but not a complete list) of commonly stored and managed data[1]:

[1]The examples mentioned are only a subset of the SIM capabilities and are valid for a SIM card. For a USIM card, some of the examples may differ from the SIM card. File names are referenced to the 3GPP TS 11.11 [1].

Table 11.1 Storage of operator information on a SIM card

Name	File name	Explanation
Service provider name	EF_ SPN	The Service Provider Name can be displayed on the handset. This file contains the name to be displayed as well as a flag indicating if the display of the name is required or not
SMS parameters	EF_SMSP	The Short Message Service header parameters can be used by the handset for mobile originated SMS. Such parameters can be the service centre address, the data coding scheme or the validity period of a message
Preferred networks	EF_PLMNsel	The entries held in this file represent the preferred roaming partners of an operator. When a handset is switched on in a foreign country, several mobile networks can be selected to roam with. If one of those networks is listed in the PLMNsel file, this network will be taken as the first choice
Forbidden networks	EF_FPLMN	In contradiction to the preferred networks, the forbidden networks represent the networks, which shall not automatically be chosen

The SIM card can also hold applications, called SIM applets, offering a certain service to the subscriber. A few examples of additional services can be found in Table 11.2.

The examples mentioned above are based on the fact that information is stored on the SIM card in dedicated files and/or in applications. Since neither operator-specific nor subscriber-specific data and services are static throughout the years of use, Over-the-Air management is necessary. A subscriber cannot be asked to consult an operator shop in order to update files or applications on the SIM card. The operator costs would be too high, and the subscriber's time would be too valuable to do so.

To be able to update information stored in SIM files Over-the-Air, "remote file management" is necessary and dedicated security features have to be applied. To delete or install SIM applications Over-the-Air, "remote applet management" processes are necessary. Chapter 3 explains the technical possibilities and details regarding Over-the-Air management of files and applets.

In order to allow a SIM application to communicate with the handset and with remote servers, the so-called SIM Application Toolkit (SAT) interface between the SIM card and the handset is necessary. This interface is briefly reviewed in the next section.

11.2.2 SIM Application Toolkit Interface SAT

The SIM Application Toolkit interface between the SIM card and the handset offers a set of commands to build applications as mentioned in Table 11.3. Using the offered

Table 11.2 Storage of subscriber-related information on a SIM card

Name	File name	Explanation
Language preference	EF_LP	The language preference can hold the preferred language of the user. The handset can read and use this language for displaying texts and menus
Subscriber's number	EF_MSISDN	The Mobile Station International Subscriber Directory Number (MSISDN) field can store one or multiple mobile numbers for the subscriber. The entry can be used by the handset to be displayed or to be used in additional services
Phone book	EF_ADN	The Abbreviated Dialling Numbers (ADN) represent the phone book on the SIM card. This phone book can be used beside of the handset-based phone book for storing the subscriber's personal contacts
Restricted phone book	EF_FDN	The Fixed Dialling Numbers (FDN) represent a phonebook, which can be used only when the handset is set into a dedicated mode. To set a handset into this mode, PIN2 is necessary. While in this mode, only numbers stored in the EF_FDN can be dialled
Last number dialled	EF_LND	The Last Number Dialled (LND) is the numbers or service strings, which the user has dialled
Text messages	EF_SMS	Short Messages (SMS) can be stored on the handset and on the SIM card. The SIM card storage can be seen as storage for more important messages, which shall not be deleted by cleaning the handset SMS buffer

commands, the SIM card can send short messages, set up a call or display menu and text items. The SIM Application Toolkit interface is specified in a GSM specification[2] [3].

A selection of the most commonly used commands can be found in the following lists:

User interface commands:

- *Set-up Menu* supplies an item list to put in Mobile Equipment (ME) Menu;
- *Display Text* displays text on screen;
- *Get Inkey* sends text on display and get character answer;
- *Get Input*sends text on display and get text answer;
- *Select Item* supplies an item list to be displayed by ME to user; and
- *Play Tone* plays a tone.

[2]In this subchapter, only the SIM card is considered. For the USIM card exists a similar specification, called 3GPP TS 31.111 [2].

Table 11.3 SIM applications offering additional services

Name	Explanation
Dynamic STK menu	Dynamic SIM Toolkit applications on the SIM card can hold a menu structure and commands to access value-added services. Value-added services can be information about the weather, sport, entertainment and much more. Since the content and the menu structure can be changed, a mechanism has to be supported to rearrange menus, to add, change or delete services
Phonebook backup	A SIM application can read the ADN file, pack it into single short messages and send it to a central OTA server. In case the subscriber has lost the handset, the SIM card is broken or the user deleted important information, the phone book can be restored from the OTA server
Handset detection (IMEI detection)	The International Mobile Equipment Identity (IMEI) represents the serial number of a handset and its capabilities. To use additional non-voice services such as GPRS (data), Wireless Application Protocol (WAP) or Multimedia Messaging Service (MMS), the handset has to be configured accordingly. Since the subscriber does not have the knowledge to configure a new handset, an automatic mechanism should be in place, where a device management system is configuring the handset accordingly. To let the device management system know that a new— not configured—handset is attached to the mobile network, a SIM application reading the serial number of the handset can be used. The SIM application checks the IMEI of the handset every time when it is switched on, and in case the number is different from any of the last numbers, a dedicated trigger is sent to the device management system
Restricted phone book	The Fixed Dialling Numbers (FDN) represent a phone book, which can be used only when the handset is set into a dedicated mode. To set a handset into this mode, PIN2 is necessary. While in this mode, only numbers stored in the EF_FDN can be dialled
Mobile TV application	Latest handsets are capable to receive Digital Video Broadcast (DVB-H) signals. Since the video message can be encrypted, a key is necessary to decrypt the video signal. This key can be stored on the SIM card. For security reason, this key can periodically be changed
Call control	When a subscriber dials a number, the handset can ask to the SIM card to verify it and if necessary change the number or the format. With such functionality, calls into a foreign network can be rerouted to a cheaper number via a partner organisation
Banking application	Banking applications are offering services such as access to personal bank account information or even executing transactions. Since the SIM card is solely controlled by the operator and offers various security functionalities, such banking applications can be stored on the SIM card

Network access commands:

- *Set up Call* initiates a call;
- *Send SMS* sends a short message to network;
- *Send SS* sends a supplementary string to network;
- *Send USSD* sends an Unstructured Supplementary Service Data (USSD) string to the network; and
- *Send DTMF* sends a Dual-Tone Multi-Frequency (DTMF) string after a call has been established.

An application such as Dynamic STK (DSTK) makes use of commands such as "Set up Menu", "Display Text" and "Send SMS" to display a menu on the handset's screen and to send the selected service to an external server. A phonebook backup application reads the file EF_ADN, packs the content into single short messages and sends them to an external server.

Using the SIM Application Toolkit interface, the SIM card can get temporary control over the handset (proactive). Figure 11.1 shows an illustration of a proactive SIM. Compared to the case where the handset sends a request to the SIM card and the SIM responds, the handset can fetch a SAT command from the SIM. The SIM card is issuing a SAT command towards the handset; the handset is reacting accordingly and sends the SIM a response back, if the command was executed successfully or not.

The 3GPP TS 11.14 [3] specification defines the SAT interface, which can be used by SIM applications, but does not define how the commands can be accessed in a unified way over various SIM card types from different SIM vendors. This means, an application written for one SIM operating system from one SIM vendor cannot be reused on another SIM operating system from another SIM vendor. This implies that the same applet has to be developed and tested for every operating system independently, resulting in high costs and additional risks.

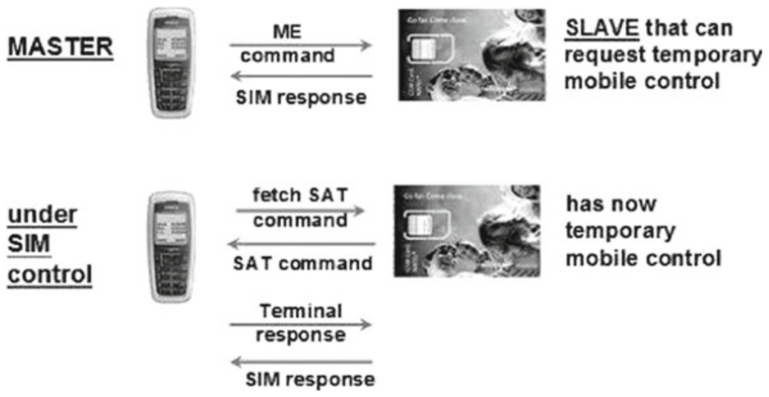


Fig. 11.1 Proactive SIM with SIM Application Toolkit

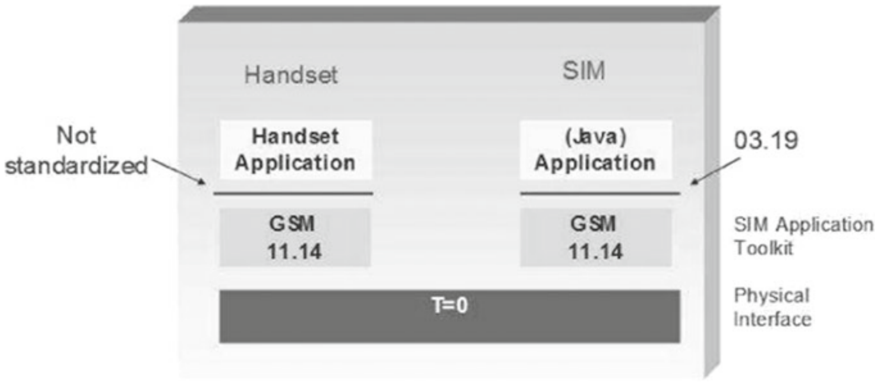


Fig. 11.2 SAT interface for Java SIM cards

Due to this fact, the Java Runtime Environment is also implemented on dedicated Java SIM cards. To be able to access the SAT functionality from Java SIM applications, the specification 3GPP TS 03.19 [4] is applied.[3] The SAT interface for Java SIM cards is illustrated in Fig. 11.2.

The 3GPP TS 03.19 [4] specification allows a service based on a Java SIM card to be developed once and to be loaded onto different Java SIM cards from different SIM vendors. Although the access to SAT functionality from Java SIM applications is standardized, reality shows that interoperability tests on every target SIM have to take place to guarantee error-free functionality.

To achieve interoperable SAT applications on Java SIM cards, a number of specifications and documents should be considered. The following list mentions the most commonly used documents:

- 3GPP TS 03.19 [4] defines the following:
  - the events for triggering a SAT applet;
  - how to access the SIM files from a SAT applet; and
  - how to implement the SAT commands ([3]) into Java classes.
- 3GPP TS 03.48 [6] defines the following:
  - Security mechanism for SAT.
- GlobalPlatform Card Specification [7] defines the following:
  - how to load, install, uninstall and delete an applet.

[3]In this section, only the SIM card is considered. For the USIM card, there is a similar specification, called 3GPP TS 31.130 [5].

11.2.3 Main Differences Between a SIM and a UICC/USIM Card

The SIM card for the GSM network contained one GSM application to authenticate the SIM towards the network. In case of a third-generation SIM card, the specifications were split into a UICC (Universal Integrated Circuit Card) and a SIM/USIM specification. Herewith, the UICC defined the base, on which a SIM application and/or a USIM application can be placed. Like this, the base for additional applications such as ISIM (IP Multimedia Subsystem SIM) [8] for new network authentication is given.

The UICC therefore allows multi-application management, where the USIM can be one of them. The USIM application as another application has an enhanced phone book supporting one name with multiple numbers assigned. Also, the file structure and the files themselves follow a USIM specification which differs from the SIM file definition.

Due to the fact that there is a split of the functionality into UICC and USIM specifications, there is not a one-to-one transition from the SIM to the UICC/USIM specifications possible. Some of the specifications were also transferred to ETSI. This is the main reason why some 3GPP and ETSI specifications are referring to each other. Furthermore, the GSM and 3GPP specifications were extended during time, resulting in multiple versions, which are bundled into Releases. For GSM specifications, for example, exists a Release 99, containing all necessary specifications to be followed by all network elements. For 3G, there exists Release 4, followed by Releases 5, 6 and 7. The Releases help to decrease interoperability problems due to the fact that for a network element, a SIM or a handset that supports a dedicated Release can be requested. On the other hand, due to the fact that multiple versions of a specification are available, one has to specify exactly the Release the implementation has to follow.

The most important specifications and their counterparts for UICC/USIM used for remote file and applet management and for SAT application design are mentioned in Table 11.4.

An overview of all specifications and Releases can be found on the 3GPP Website: http://www.3gpp.org/specs/numbering.htm

11.3 OTA—Over-The-Air Management

As shown in Sect. 11.2.1, there exist various cases where Over-the-Air management of files and applets is necessary. The most commonly used transport bearer for remote SIM management is Short Messages (SMS). Commands, the so-called APDU (Application Protocol Data Unit), are encrypted and packed into SMS. The OTA server sends the SMS to the SMS Centre which is responsible for the delivery to the handset. In case a handset is not reachable, the SMS Centre is responsible for retrying to deliver the SMS until a certain time is reached, the “validity period”.

Table 11.4 Relation of main GSM and 3GPP/ETSI specifications

Name	GSM specification (Rel. 99)	3G/GSM specification (>Rel.4)
Security mechanism	3GPP TS 03.48 [6]	3GPP TS 23.048 [9]
SIM/UICC specification	3GPP TS 11.11 [1]	ETSI 102 221 [10] (UICC definition) 3GPP TS 31.102 [11] (USIM definition) 3GPP TS 51.011 [12] (SIM definition)
SIM Application Toolkit (U)SAT	3GPP TS 11.14 [3]	3GPPTS 31.111 [2]
Java API for (U)SAT	3GPP TS 03.19 [4]	ETSI 102 241 [13] UICC 3GPPTS 31.130 [5]

Table 11.5 3GPP TS 03.48 [6] security mechanism

Name	Security mechanism applied
Message authentication	Cryptographic checksum or a digital signature
Message integrity	Cryptographic checksum
Message confidentiality	Message ciphering
Replay detection	Counter
Proof of receipt	Status code packed into response message

When the SMS is delivered to the handset, the handset recognizes according to the "message class" whether the SMS has to be handled by the handset or by the SIM card. In case of a Message Class 2, the SMS is handed over to the SIM card. The card decrypts the message and executes the included commands according to the specification 3GPP TS 11.11 [1]. Examples of commands to be executed by the SIM card are as follows:

† *Select*	selects the file with the given address
† *Read Binary*	reads the file type transparent
† *Update Binary*	updates a transparent file
† *Read Record*	reads one record of the file type linear fixed
† *Update Record*	updates a record of a linear fixed file
† *Envelope*	transfers 3GPP TS 11.14 coded data to the SAT application

In order to prevent any other party than the operator remotely managing a SIM card, security (according to the specification 3GPP TS 03.48[4]) [6] is applied on top of the file and applet commands mentioned above. The security mechanisms are applied between the OTA server and the SIM card. The main security features defined in the 3GPP TS 03.48 [6] specifications are summarised in Table 11.5.

[4]In this section, only the SIM card is considered. For the USIM card, there exists a similar specification, called 3GPP TS 23.048 "Security mechanisms for the SIM application toolkit" [9].

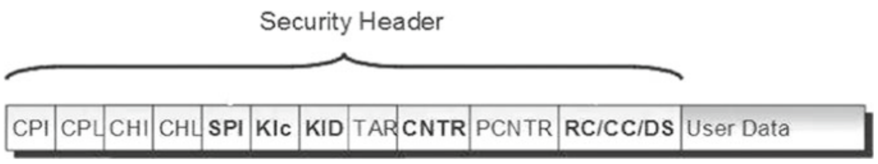


Fig. 11.3 OTA security header

In order to apply security mechanisms to remote file or applet messages, a security header is included into the SMS, illustrated in Fig. 11.3. The security header defines which of the above-mentioned security mechanisms are applied. The Security Parameter Indicator (SPI) defines the security applied to the message. The Ciphering Key Identifier (KIc) is defining the algorithm using a symmetric key—stored on the OTA server and on the SIM card—to cipher the message. Since multiple keys can be pre-stored in the U(SIM) and can be used for the ciphering, KIc also defines which key number shall be used by the encryption algorithm. Key Identifier (KID) represents the algorithm, also using a symmetric key, to calculate the cryptographic checksum and the digital signature. Also, KID is additionally defining which key number to be used by the encryption algorithm. CNTR is the counter to detect, whether a message is received more than once. The counter has to be set in the security header. For every message sent to the SIM card, the counter is increased by the OTA server. The SIM card checks towards the counter stored on the SIM, whether the counter was increased compared to the last successfully received message. If the counter received from the OTA system is below or equal to the counter on the SIM card, the received message is seen as a repetition of a previous message and is discarded. Figure 11.4 illustrates OTA security mechanisms.

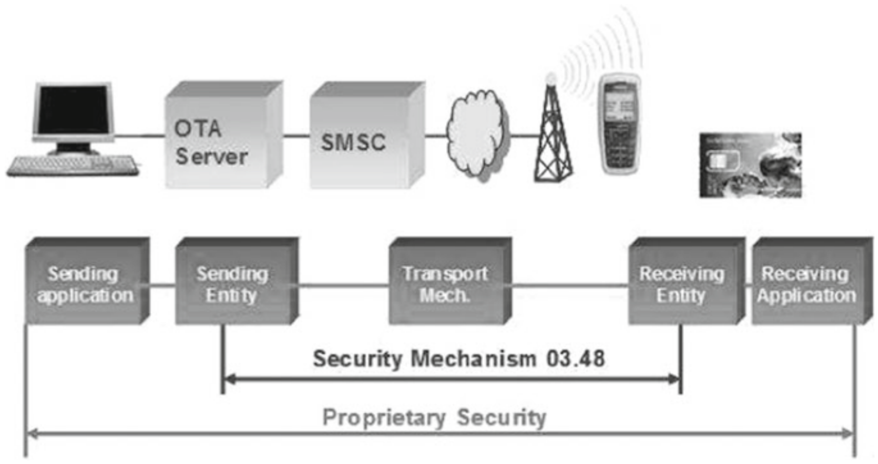


Fig. 11.4 Security mechanisms in OTA management

During the definition phase of a SIM card profile, it is agreed between the operator and the SIM vendor, whether the SIM supports one or multiple keys for ciphering, cryptographic checksum and digital signature. According to the 3GPP specifications, up to 16 keys can be used.

11.3.1 OTA Server Capabilities

In case of Over-the-Air management, the extensive security features of a SIM card need to be supported as well on the OTA server side. The OTA keys for ciphering, digital signature and cryptographic checksum (KIc, KID) have to be stored on the OTA platform in a secure way, protected from any misuse. The OTA keys can be used by different encryption algorithms, according to the specification 3GPP TS 03.48 [6]. Before this specification was in place, every SIM vendor provided their own proprietary algorithms. Since SIM cards are in use for a long time, such proprietary SIM cards are still in operation. A sophisticated OTA server therefore needs to also support the proprietary algorithms from different SIM vendors.

In addition to the security aspects of a SIM card, an OTA server also needs to know how to build correct APDUs and SMS with the header and security header according the 3GPP specifications.

Since the various SIM cards on the market support different features and follow different Releases of the 3GPP specifications, the OTA server has to know the capabilities for every SIM card. The capabilities are summarized in a "SIM Profile", and the OTA server needs to offer the functionality to configure the capabilities for every SIM stored in the OTA system. By defining a SIM Profile, the server knows which files exist on the SIM card. Furthermore, additional information such as the path, where the file is located, the length or the file type is defined. The same information has to be stored on the OTA server in order to create correct APDU commands. In addition, to prevent message duplication, counters accompany the APDU commands and are checked by the SIM card for correctness. For every new message, the counter has to be increased. Messages containing a too low counter will be ignored by the SIM card. The OTA server needs to manage the correct value of the counters and include within the messages.

The SIM Profile also defines the remote applet management capabilities. To successfully execute remote applet management, not only do the applets have to be stored on the OTA server, but also various parameters according to the 3GPP TS 11.11 [1] and GlobalPlatform Card Specification. The parameters are used to create correct remote applet download messages.

The starting point of an applet download is a Java application, which has to be converted using the standard Sun tools into a ".cap-file". The ".cap-file" has to be transformed into a ".jcl-file", which is downloaded by the OTA server according the Open Platform Card Specification. The applet can have different states on a SIM card. It can be loaded, but not yet visible to the user (status: installed), or it can be installed and visible to the user (status: selectable). For the installation, therefore, two different commands are necessary. Similarly, the removal can be handled in two steps, one for removing the visibility to the user or a complete removal from the SIM card.

The above-mentioned capabilities are essential in order to achieve successful remote file and applet management of SIM cards, as used for the cases mentioned in Sect. 11.2.1.

Since some of the remote file and applet management activities are done in an active mode, and some in a reactive mode, the OTA server needs to support both modes. Active remote management is mainly used to configure a whole subscriber base or a subset of it with new or changed data. Active mode means also to update several ten or hundred thousand of SIM cards in a certain time frame. For every target SIM, the batch processing mechanism calculates the necessary APDU, applies the security and packs it into messages. The OTA server also has to handle the delivery of every single message to the SMS Centre and track the success of the delivery. Every target SIM can be of a different SIM profile where different securities have to be applied. In any case for every SIM, the OTAkeys are different and every single SMS has to be encrypted independently. The OTA server has to provide such calculation capabilities and performance to reach an efficient remote update of the target SIM cards.

Compared to batch processing, the reactive mode is dealing with a single SIM card at a time. More important for the reactive mode is that any user requests are immediately handled by the OTA server and an appropriate reaction has to be sent to the requesting SIM card. An example for this mode is the request of a user to backup their phone book. Or in case of dynamic STK menus, the user can have the possibility to add or delete services themself. In this case, the user is waiting for an immediate response, which the OTA server needs to handle immediately.

11.4 Limitations and Improvements

Remote management of files and applets on a SIM card can require from a few commands up to several kilobytes of data to be transferred Over-the-Air. Applets usually have a size from some kilobytes up to 20 kilobytes or even more. In an operators environment, applet download is performed for sizes up to 5 or 6 kilobytes. It is only in test environments that larger applets are practicable to be downloaded.

The main reason for this limitation is the available bearer. The only reliable bearer today is the short message SMS. Since an SMS has a size of 140 Bytes and due to header and checksum information, only a part of an SMS is containing data. The data part of an SMS is called payload. There exists SMS with reduced header information, so-called concatenated SMS, allowing the payload to be increased slightly.

The more data that has to be downloaded, the more SMS have to be created, managed and sent. The more SMS that have to be sent to a single subscriber, the higher the risk of an incident. Heavy SMS load results in longer distribution time, which can be interrupted especially while changing location. Some of the necessary

SMS might be not delivered in the correct order which might entail complicated retry mechanisms for some or even all the messages.

Due to those risks, large data downloads require alternative bearers to SMS. There exist some faster bearers, but due to several reasons, these have so far seen limited use in the operator's environment.

Direct SS7 Access:
Instead of using the operators SMS Centre, direct access to the signalling network (SS7) can be established. Direct SS7 access can be faster than using the operators SMS Centre, because the performance does not depend on the SMS Centre, which might be temporarily loaded from other systems. By bypassing the SMS Centre, an increase of 10 to 20% in distribution time might be reached; however, the commands still have to be packed into single SMS and the ordering and retry problems still exist.

SMS Cell Broadcast (CB):
Cell Broadcast messages are generated once and distributed to all SIM cards within an area. The configuration of every SIM card determines whether a CB message shall be treated by the receiving SIM card or not. Cell Broadcast messages save time in the message creation since the same message is valid for all target SIM cards. Cell Broadcast messages have a similar size to SMS; therefore, the problems with the ordering and retry policy also exist here. Due to the fact that one message has to be valid for all target SIM cards, the same security has to be applied for all SIM cards. This implies a higher risk. If an OTA key were to be hacked, the same OTA key could be used for all SIM cards, and this represents a high risk that no operator is willing to take.

Unstructured Supplementary Service Data (USSD):
Current SIM cards in the operators' use can send USSD strings, but not yet receive them. The main advantage of USSD is that the data is transferred in one of the signalling channels, which makes it fast. The package size on the other hand is similar to SMS. USSD has no store capability, which makes retry capabilities necessary in case the subscriber is not reachable. The problems with the ordering and its retry exist as for the direct SS7 access. USSD as a bearer might become more interesting, as the receiving of USSD data directly by the SIM card is now in the latest 3GPP standards. On the other hand, experience has shown that the widespread handset support of new features might take a long time. USSD might not be needed in the future since there is another solution with BIP/CAT_TP now slowly emerging.

Bearer Independent Protocol BIP with CAT_TP:
The Bearer Independent Protocol BIP with its overlaying Card Application Toolkit Transport Protocol CAT_TP has been in the 3GPP standards for several years. But it is only recently that the handset vendors have implemented BIP into their handsets. The SIM vendors on the other hand have supported BIP for several years.
BIP allows the opening of a data channel from the handset to the OTA server and to the SIM card, resulting in an end-to-end data channel. BIP message flow is illustrated in Fig. 11.5. Compared to SMS, where one SMS has the size of 140 Bytes, a data

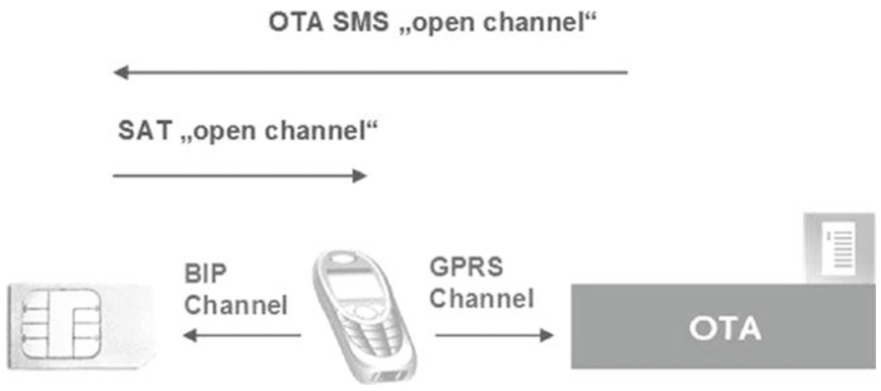


Fig. 11.5 BIP message flow

package for BIP contains 1472 Bytes. Not only is the package size bigger, but also the transport—by using GPRS or any advanced data channel from the handset—is much faster. BIP was designed not only to have a fast access channel towards an OTA server, but also local bearers such as Bluetooth or infrared would be possible. A data channel via BIP can be opened by the SIM card by sending an appropriate "open channel" command to the handset. When the OTA server wants to open a data channel, the command has to be packed into an SMS, which is sent from the OTA server to the SIM card. As soon as the data channel is established, the SIM card can send and receive data packages. CAT_TP is the overlaying protocol to ensure transport security and is based on UDP. Alternative solutions using TCP instead of CAT_TP are also studied. The market has not yet shown, in which implementation will succeed. BIP is defined in the 3GPP TS 31.111 [2], CAT_TP in the ETSI TS 102 124 [14] and TS 102 127 [15].

11.4.1 Customer-Managed Applications

Due to the security features that a SIM card is offering, the OTA system requires access to the OTA keys for remote file and applet management. Since the SIM card belongs to the operator, no party other than the operator has access to the OTA keys and can operate changes on a SIM card.

New services could be provided by third-party service providers, but the operator remains the only party to be able to download additional applets to the SIM card. A dedicated restricted access to third-party service providers could be granted by the operator, but it is the operator's responsibility to ensure that the applet is reliable and secure and does not open security holes.

In all known solutions, where a third party is involved in providing additional services, the operator is still heavily involved. Due to this fact, such implementations are rare and high costs for the operator and the service provider are the result. The applet, as well as the interfaces, has to be reliable and secure, requesting well-proven processes.

Since there are also legacy SIM cards in the field, the new services are not applicable to all SIM cards. In order to address the whole customer base with a new service, SIM card exchange might even be necessary, resulting in additional costs. Finally, the memory management on SIM cards from different vendors is not identical and might cause additional problems.

Future SIM cards are more powerful and provide more memory and so the number of customer-managed solutions might increase. On the other hand, the increase in performance and memory on the handset is much greater than for the SIM, and due to the shorter life cycle of a handset, it is also deployed much faster. The future of customer-managed solution on the SIM card is therefore not yet completely clear.

11.5 The Smart Card Web Server

The Smart Card Web Server (SCWS) is a SIM-based application development and execution environment that uses Web technologies. The SCWS is effectively a small web server on the SIM card, which can offer static and dynamic content to the user via the normal phone browser. Benefits of this approach include an improved graphical user interface for SIM-based applications and trustworthy content due to the attack resistance of the SIM card. It offers a different OTA approach to overcome some of the technological limitations discussed in Sect. 11.4. As an example, SCWS can be used to run applications and services that can be remotely managed by an OTA platform using a web protocol such as HTTPS [16], noting that in OTA management mode, the SCWS may act as the client, instead of its operational server role. The updates can be managed and coordinated by a Remote Administrative Server (RAS), giving the ability for the operator to manage user-specific applications efficiently via widely used Internet protocols [17].

To secure the Web-based communication, the SCWS uses the following mechanisms [16].

- User authentication mechanisms as defined in HTTP;
- HTTP over TLS (HTTPS) for security protection; and
- Access Control Policy (ACP) as defined in Open Mobile Alliance (OMA) [18].

A RAS may use either the Lightweight Administrative Protocol (LAP) or the Full Administrative Protocol (FAP) for remotely updating and modifying content. Both of these protocols have been standardised.

11.5.1 Lightweight Administrative Protocol

This protocol is used when the data that needs to be send by OTA is small enough to fit within a few text messages. It uses SMS with encapsulated HTTP commands. The formatting of the SMS messages is carried out adhering to ETSI TS 102 225 [19]. The protocol is optional in OMA recommendations, so it is unlikely to be supported in many implementations, especially as OMA specifies quite a large minimum size (512 bytes) for OTA administrative command messages [17].

11.5.2 Full Administrative Protocol

The Full Administrative Protocol can handle even large OTA messages and initially sets up a tunnel connection between the SCWS and the administrative entity [16]. This protocol uses the widely deployed and proven HTTP protocol, and is readily supported via an operator's network [17].

11.6 SIM Lifecycle Management

Over-the-Air management of a SIM card is used when the SIM card is active and in use by the customer. Since exchanging customers' SIM cards due to malfunction or missing features is very cost intensive, the necessary features have to be planned and tested carefully. In order to successfully launch new SIM cards with new services, a process needs to be followed. The process is illustrated in Fig. 11.6.

Planning:

Before a SIM card can be ordered by an operator, the card profile has to be defined. This contains the electrical capabilities "the electrical profile", the security capabilities, but also the graphical profile with the definition of all general and individual card printing.

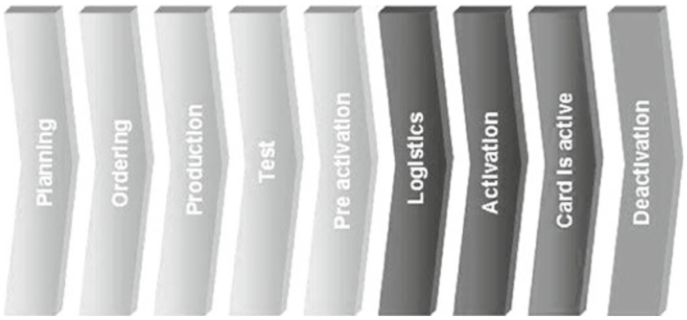


Fig. 11.6 The SIM lifecycle management process

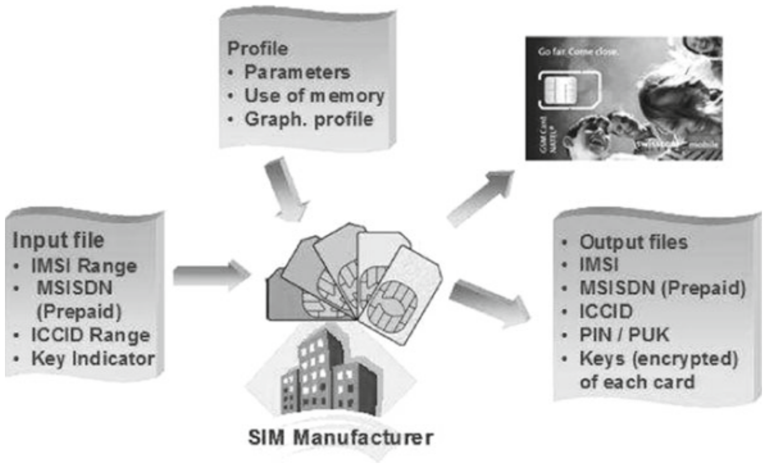


Fig. 11.7 SIM production—Illustration 1

Ordering:
The operator orders SIM cards from a vendor using an “Input File”. The Input File contains the profile information as well as SIM card’s serial number (Integrated Circuit Card Identifier (ICCID)) and the International Mobile Subscriber Identity (IMSI). In some dedicated cases, also the Mobile Station International Subscriber Directory Number (MSISDN) might be assigned already (possible in case of prepaid cards).

Production:
During production and personalisation, the SIM manufacturer creates all secret data (PIN codes, OTA keys) and writes them into the card’s memory. An “Output File” is generated, containing all necessary data to be loaded into the operator’s network entities. SIM production is illustrated in Figs. 11.7 and 11.8.

Test:
In order to ensure that the SIM will properly work when shipped to the customer, and also to ensure that later Over-the-Air management is possible for fulfilling future changes, the SIM card has to be extensively tested. The costs of replacing a set of several thousand SIM cards already shipped to the customers are many times more expensive than ensuring its correctness during the test phase.

Pre-Activation:
The pre-activation phase is where the output file data has to be loaded into the operators’ network systems in a secure manner. Systems to be issued with SIM data include the HLR (Home Location Register), AuC (Authentication Centre), CCBS (Customer Care and Billing Systems) and OTA systems. Even after this process, the card is still not active.

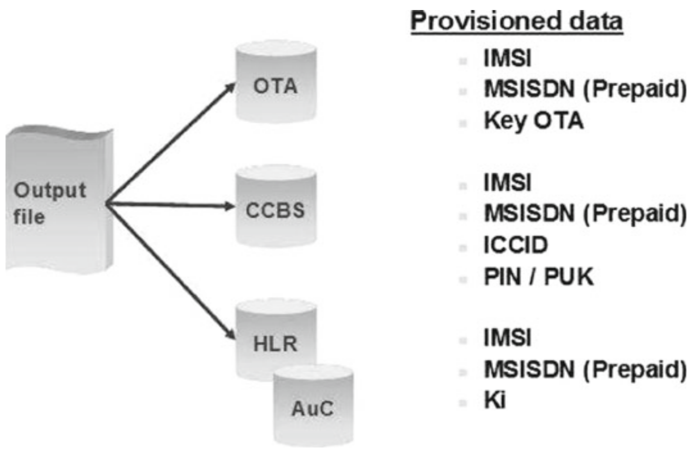


Fig. 11.8 SIM Production—Illustration 2

The process from the planning to the pre-activation can be supported by dedicated systems, in order to avoid manual provisioning and human errors. Systems supporting this phase are sometimes called Resource Management Systems.

Logistics:
The SIM cards are shipped from the vendor to the operator and distributed to the operator's shops.

Activation:
When the SIM card is shipped or handed over to the customer, the MSISDN is assigned and the card is activated in all systems, where the card data was provisioned. Many operators have a specific customer activation tool in their outlet that has an online connection to their customer management system, which allows card activation once the subscriber has obtained the card.

Card is active:
The SIM card is active and in use. Changes or additional services are managed remotely Over-the-Air by an OTA system.

Deactivation:
The SIM card has no validity period as bank cards have. There are still cards in use which are more than 10 years old. Many operators worry about exchanging those SIM cards due to logical obstacles, low customer acceptance and high costs. Beside the planned exchange of a SIM card by the operator, also a lost or stolen SIM or customer churn defines the end of life of a card. The SIM card is deactivated in all systems, where the card data was provisioned, but usually the card data remains in the systems in order to be able to track its history.

11.7 OTA Provisioning for Embedded SIM

It must be noted that not all SIMs are installed in mobile phones. More and more devices are now being connected to the Internet to form part of the Internet of Things (IoT), and embedded SIMs can be used to establish Machine-to-Machine (M2M) Communication. These embedded SIMs (eSIM) are not always physically accessible, or they may not be removable if soldered into the device. Therefore, remote Over-the-Air provisioning and management of the embedded SIM is a vital aspect in the M2M market.

As the number of devices with embedded SIMs increase, so does the requirement to manage them. The device security requirements are no less than conventional removable SIMs and indeed may be greater for critical infrastructure. OTA provisioning and management are required in embedded SIM, M2M scenarios, and this has been standardized by the GSMA in the embedded SIM specification [20].

In a competitive M2M market space, an embedded SIM needs the ability to change Mobile Network Operators (MNOs). In order for a machine equipped with an embedded SIM to change its MNO conveniently and securely, a M2M security model has been introduced.

11.7.1 M2M Security Model

The new embedded SIM architecture includes two key network elements for a SIM to select and install a Mobile Network Operator (MNO), as illustrated in Fig. 11.9.

The first element is the Subscription Manager Data Preparation (SMDP), which securely encrypts the operator credentials ready for OTA provisioning. It also manages the installation of these credentials onto the eSIM [20].

The second element is the Subscription Manager Secure Routing (SMSR), which securely delivers the encrypted operator credentials to the SIM. This also facilitates remote management of the eSIM thereafter which includes loading, enabling, disabling and deleting profiles [20]. Once the MNO is established, the embedded SIM can use the methods for OTA communication and management as described previously.

11.8 In Conclusion

The SIM card with its security features can be used for various services, such as storing the phone book, managing the roaming list or configuring operator-specific parameters. These services are kept in files, present on the SIM card, which are specified by the ETSI and 3GPP standardisation bodies. Additional services can also be added by loading or modifying applications (applets) on the SIM card. Providing

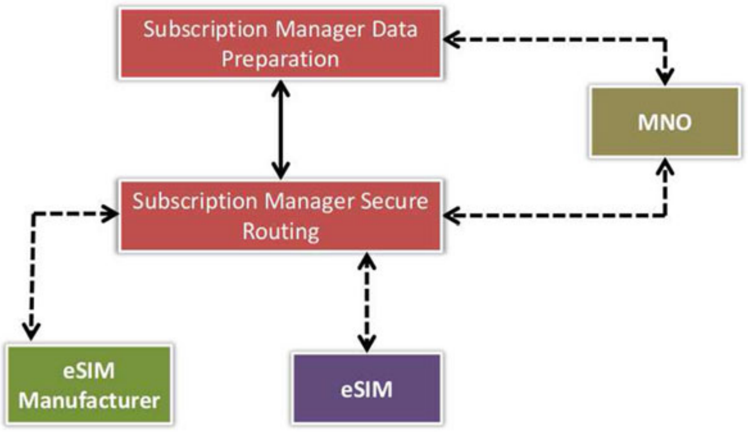

Subscription Manager Data Preparation
MNO
Subscription Manager Secure Routing
eSIM Manufacturer
eSIM

Fig. 11.9 M2M MNO selection architectures: based on [20]

menus to access value-added services via DSTK menus, backing up the phone book on the SIM card and configuring mobile TV applications are examples of such SIM card applications.

Applets, in particular, can provide functionality to the subscriber that requests their interaction. Displaying data or requesting the input of subscribers data is based on functionality provided by a dedicated interface between the SIM card and the handset. This is known as the SIM Application Toolkit (SAT) interface. In order to be able to load the same applet on several SIM cards, Java applets are used. To unify the access by the Java applet to the SAT functionality, a dedicated specification provided by ETSI/3GPP is available. Nevertheless, interoperability tests between different SIM cards are necessary to guarantee error-free functionality.

Due to the variety of services provided by and due to the long life cycle of the SIM card, remote Over-the-Air (OTA) management is essential. Dedicated OTA systems exist to support this stage of the SIM life cycle. Whether single or a whole batch of SIM cards are being updated Over-the-Air, the OTA systems need to follow certain rules for coding the messages, adding security to them and finally managing the messages until they are successfully executed on the SIM card. The coding of the messages and the security to be applied is well defined by ETSI/3GPP specifications. Due to the security features offered by the SIM card, it is seen as a trusted platform offering the storage and management of personal and sensitive data. On the other hand, to cope with the features offered by a SIM card, the OTA system needs to be able to handle for every single SIM card type, its capabilities and its dedicated security features. Despite the wide standardization of SIM cards, the OTA system needs to be able to follow different versions of the standards to be able to execute remote file and applet management on a wide range of SIM cards successfully. Coding can differ from one SIM type to another, different security features might be applied and even proprietary security features still have to be supported.

Most of the remote Over-the-Air management in today's mobile telecom environment is made by sending Short Messages (SMS) to the SIM card. Short Messages have a limitation in terms of speed and bandwidth. Therefore, alternatives such as the Bearer Independent Protocol (BIP) are being specified and are starting to be used in order to replace Short Messages. The SCWS is designed around the reuse of more conventional Internet communication protocols, although as a technology SCWS has not found widespread use.

Due to the long life cycle of a SIM card and due to legacy performance limitations, the issuing of new SIM cards and its services have to be planned carefully. To keep control of the features when a SIM card is actively used, an OTA system is essential. To be able to manage the whole life cycle of a SIM card from the planning to the deactivation, a SIM lifecycle management system is needed.

The SIM card and its Over-the-Air management have been well standardized/specified by a range of organizations such as 3GPP, ETSI, OMA, GSMA and the SIM Alliance. However, the number of players and competitors in this space is increasing as we move towards M2M systems and the IoT in general. Nevertheless, OTA and SIM lifecycle management systems remain essential for provisioning and managing the life cycle of SIM cards and embedded SIM equivalents.

References

1. 3GPP TS 11.11 Specification of the Subscriber Identity Module - Mobile Equipment (SIM-ME) Interface. More information at http://www.3gpp.org/DynaReport/1111.htm, accessed August 2016.
2. 3GPP TS 31.111 Universal Subscriber Identity Module (USIM) Application Toolkit (USAT). More information at http://www.3gpp.org/DynaReport/31111.htm, accessed August 2016.
3. 3GPP TS 11.14 Specification of the SIM Application Toolkit (SAT) for the Subscriber Identity Module - Mobile Equipment (SIM-ME) interface. More information at http://www.3gpp.org/DynaReport/1114.htm, accessed August 2016.
4. 3GPP TS 03.19 Subscriber Identity Module Application Programming Interface (SIM API) for Java Card. More information at http://www.3gpp.org/DynaReport/0319.htm, accessed August 2016.
5. 3GPP TS 31.130 (U)SIM Application Programming Interface (API); (U)SIM API for Java Card (TM). More information at http://www.3gpp.org/DynaReport/31130.htm, accessed August 2016.
6. 3GPP TS 03.48 Security mechanisms for SIM application Toolkit; Stage 2. More information at http://www.3gpp.org/DynaReport/0348.htm, accessed August 2016.
7. GlobalPlatform Card Specifications. More information at http://www.globalplatform.org/specificationscard.asp, accessed August 2016.
8. 3GPP TS 31.103 Characteristics of the IP Multimedia Services Identity Module (ISIM) application. More information at http://www.3gpp.org/DynaReport/31103.htm, accessed August 2016.
9. 3GPP TS 23.048 Security mechanisms for the (U)SIM application toolkit; Stage 2. More information at http://www.3gpp.org/DynaReport/23048.htm, accessed August 2016.
10. ETSI TS 102 221 Smart Cards; UICC-Terminal interface; Physical and logical characteristics. More information at http://www.etsi.org/deliver/etsi_ts/102200_102299/102221/08.02.00_60/ts_102221v080200p.pdf, accessed August 2016.

11. 3GPP TS 31.102 Characteristics of the Universal Subscriber Identity Module (USIM) application. More information at http://www.3gpp.org/DynaReport/31102.htm, accessed August 2016.
12. 3GPP TS 51.011 Specification of the Subscriber Identity Module - Mobile Equipment (SIM-ME) interface. More information at http://www.3gpp.org/DynaReport/51011.htm, accessed August 2016.
13. ETSI TS 102 241 Smart Cards; UICC Application Programming Interface (UICC API) for Java Card (TM). More information at http://www.etsi.org/deliver/etsi_ts/102200_102299/102241/09.01.00_60/ts_102241v090100p.pdf, accessed August 2016.
14. ETSI TS 102 124 Smart Cards; Transport Protocol for UICC based applications; Stage 1. More information at http://www.etsi.org/deliver/etsi_ts/102100_102199/102124/06.01.00_60/ts_102124v060100p.pdf, accessed August 2016.
15. ETSI TS 102 127 Smart Cards; Transport Protocol for CAT applications; Stage 2. More information at http://www.etsi.org/deliver/etsi_ts/102100_102199/102127/06.06.00_60/ts_102127v060600p.pdf, accessed August 2016.
16. Smart Card Web Server: How to bring operators applications and services to the mass market, SIMAlliance, http://simalliance.org/wp-content/uploads/2015/03/WP_SIMallianceSCWS_Feb09_Final.pdf, accessed August 2016.
17. Smart Card Web Server Stepping Stones, OMA, SIMAlliance, http://simalliance.org/wp-content/uploads/2015/03/SCWS_SteppingStones_2009_v1.0.01.pdf, accessed August 2016.
18. OMA Smart Card Web Server, Version 1.2, http://technical.openmobilealliance.org/Technical/technical-information/release-program/current-releases/smartcard-web-server-v1-2, accessed August 2016.
19. Interoperability Stepping Stones Release 7, SIMAlliance, http://simalliance.org/wp-content/uploads/2015/06/SteppingStones_R7_v100.pdf, accessed August 2016.
20. Remote Provisioning Architecture for Embedded UICC Technical Specification, Version 3.1, 27 May 2016, http://www.gsma.com/connectedliving/wp-content/uploads/2016/07/SGP.02_v3.1.pdf, accessed August 2016.

Chapter 12
Smart Card Reader and Mobile APIs

Damien Sauveron, Raja Naeem Akram and Konstantinos Markantonakis

Abstract The aim of this chapter is to describe the main middlewares and APIs used to manage and access smart card readers and smart cards, along with the mobile phone APIs to access smart cards, SIM card and Secure Elements. It is illustrated by samples of code that the reader of this book will be able to reuse to quickly develop his/her first applications to communicate with smart cards.

Keywords Smart card reader APIs · Mobile phone APIs · PC/SC · Wrappers · JSR 268

12.1 Terminology: Smart Card Reader, IFD, CAD and Terminal

Smart cards are becoming increasingly prevalent and in order to develop a host card-aware application it is fundamental to manage the readers enabling smart card communication.

A smart card reader is also known as a card programmer (because it can read data from and write data to a card), a card terminal, a Card Acceptance Device (CAD) or an InterFace Device (IFD). There is a slight difference between the card reader and the terminal. The term "reader" is generally used to describe a unit that interfaces

D. Sauveron (✉)
XLIM UMR CNRS 7252, Université de Limoges, Limoges, France
e-mail: damien.sauveron@unilim.fr

D. Sauveron
LaBRI UMR CNRS 5800, Université de Bordeaux, Bordeaux, France

R.N. Akram · K. Markantonakis
Smart Card Centre, Information Security Group,
Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK
e-mail: r.n.akram@rhul.ac.uk

K. Markantonakis
e-mail: k.markantonakis@rhul.ac.uk

K. Mayes and K. Markantonakis (eds.), *Smart Cards, Tokens, Security and Applications*, DOI 10.1007/978-3-319-50500-8_12

with a host for the majority of its processing requirements. In contrast, a “terminal” is a self-contained processing device.

As already shown in this book, smart cards are portable data cards that must communicate with an external device to gain access to a display device or a network. Thus, they can be plugged into a reader, commonly referred to as a card terminal, or they can operate using Radio Frequencies (RF).

The reader provides a path for an application to send commands to the card and receive responses from it. There are many types of reader available, and the easiest way to describe a reader is by the way it is interfaced to a host: serial RS232 connection; Personal Computer Memory Card International Association (PCMCIA); Universal Serial Bus (USB); ExpressCard; floppy disk slots; parallel ports; Infrared Data Association (IrDA) connection; Bluetooth connection; keyboard wedge readers; built-in reader; etc.

Readers have many form factors and capabilities. For example, some card readers come with advanced security features such as secure PIN entry, secure display or an integrated fingerprint scanner for the next generation of multilayer security and three-factor authentication.

A mobile phone can be considered by several aspects as a “terminal” that can communicate with smart cards or secure elements. It can communicate with Subscriber Identity Module (SIM) cards and even with contactless cards when a Near Field Communication (NFC) solution is integrated.

As illustrated later in this chapter, card readers can easily be integrated into a host running one of common operating systems (from Windows 98 to at least Windows 10, Linux, Unix, FreeBSD, MAC OS X).

To summarise the main steps of the communication between the host application and a card:

- First, the application has to communicate with the reader connected to the host. It is worth noting that in the past, most reader manufacturers used their own low-level protocol for communication between the host and its reader. Fortunately, in early months of 2000, USB Implementers Forum [1] launched an initiative to standardise it with the Chip Card Interface Device (CCID) protocol which will be presented later in this chapter.
- Second, the reader communicates with the card, acting as the intermediary before sending the data to the card using a low-level protocol (usually T = 0, a protocol byte-oriented, or T = 1, a protocol block-oriented, or even a contactless protocol) supported by the related card. Fortunately, the ISO groups have proposed an upper level protocol with is standard independent of the low-level layer for communication with a smart card: the APDU (Application Protocol Data Unit) format. In short, the low-level protocols transport the APDU commands and responses.
- Third; the card will process data and return a response (an APDU response) to the reader, which will then return the data to its originating source (i.e. the host application).

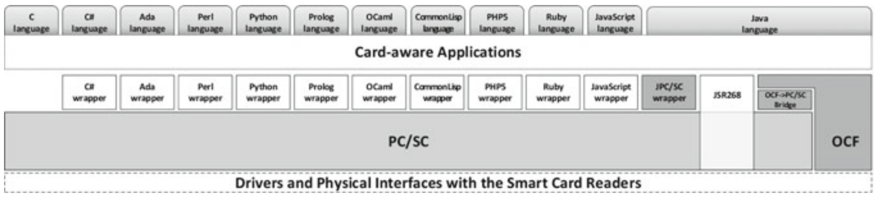


Fig. 12.1 The main stacks to communicate with a reader from a host

If there is no one-size-fits-all approach to smart card reader communication, in order to facilitate the development of the host application using the smart card, several software stacks to abstract these communication protocols and the management of the readers have been proposed. The main stacks (OpenCard Framework (OCF), JSR268 and PC/SC) to communicate with a reader from a host are shown in Fig. 12.1. They are the topics of the following sections.

OpenCard Framework [2–4] and JSR268 (so-called Java Smart Card I/O API) [5] make it possible to access readers on all systems supporting Java, whereas PC/SC provides access to the card readers through various programming languages. Before going deeper in the presentation of these frameworks, it can be mentioned that OCF was the first framework for Java and that even though its development has been stopped, it can still be used to develop applications. JSR268 is the successor of OCF and it has the advantage to be natively supported in the Java Runtime Environment (JRE). However, JSR268 provides fewer services, so studying OCF can be inspiring to develop additional services. It can also be noted that both OCF and JSR268 can be used on top of PC/SC. As explained in [6], if PC/SC and OCF share some similar concepts, they are two orthogonal standards, and this discussion can be extended to PC/SC and JSR268.

As aforementioned, mobile phones are a kind of terminal with their own operating systems (Android, iOS, etc.), they do not support the same stacks. In Sect. 12.5, some APIs and solutions to communicate from mobile phone to smart card and similar secure element will be presented.

12.2 OCF: OpenCard Framework

OpenCard Framework is an open standard framework that provides interoperable smart cards solutions written in Java across many hardware and software platforms. Its architecture and its set of application program interfaces (APIs) enable application developers and service providers to build and deploy smart card-aware solutions in any OpenCard-compliant environment.

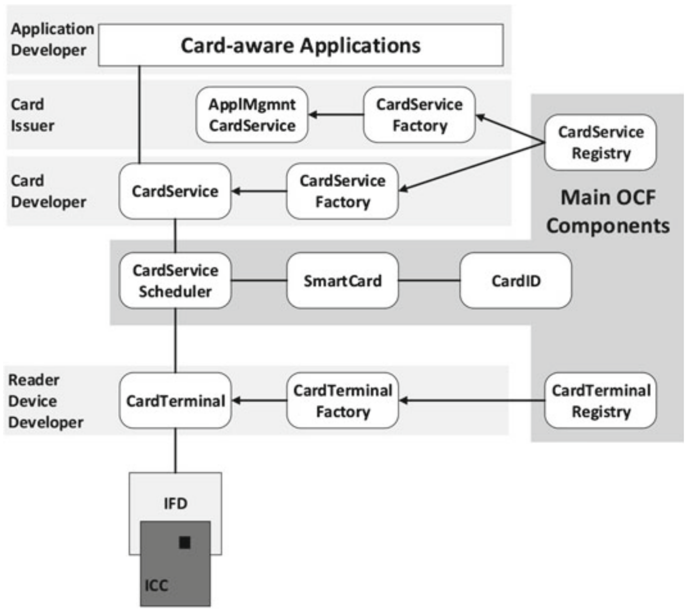


Fig. 12.2 The OpenCard Framework (OCF) architecture

12.2.1 Overview

The OCF project was initiated in 1997 by the OpenCard consortium which comprised Gemplus, Giesecke&Devrient, IBM, Schlumberger, Sun microsystems, Visa International, and others. The OCF makes it possible to access readers, smart cards and their applications in Java. The reference implementation of OCF, as presented in Fig. 12.2, includes a set of Java classes, packages and APIs that contains all the necessary functionalities. The OCF architecture defines the `CardTerminal` and `CardService` classes and offers a standardised high-level interface to applications.

If the reader manufacturers want to make their readers available to applications, they just need to provide a `CardTerminal` class, which encapsulates reader behaviour, and a `CardTerminalFactory` class which has to be registered with the `CardTerminalRegistry`. It keeps track of all readers to the OpenCard Framework, and it is used by the framework to create `CardTerminal` instances when the framework is initialised.

Card services offer smart card functionality to application developers via high-level interfaces. Thus, card manufacturers have to provide `CardService` classes encapsulating the behaviour of their smart cards and a `CardServiceFactory` class which must be registered with the `CardServiceRegistry` (thereafter used by the framework to instantiate card services). For example such a service could be a `FileSystemCardService` for the cards hosting a file system. By the way, there is one default implementation of such an ISO file system provided in the optional package of OCF (`opencard.opt.iso.fs.FileAccessCardService`).

With this framework, application developers mainly use card services already offered by the framework. However, if they would like to propose their own smart card application to other developers they can also develop their own service. The example presented in the next section will illustrate how it is possible to communicate with the card and thus develop a new service. OCF is described in much greater detail within the reference [3]. However, it should be noted that the main problem with OCF is that maintenance and development was stopped in February 2000.

12.2.2 Example

Please Note: All the examples in this chapter assume that there is on the card a default selected application with a Master File (MF, i.e. 3F00) and that the host application tries to select it.

The file `opencard.properties` should be configured as in Listing 12.1

```java
package samples;

import opencard.core.service.SmartCard;
import opencard.core.service.CardRequest;
import opencard.core.terminal.*;
import opencard.opt.util.*;

public class SampleAPDUSent {
   public static void main(String[] args)
   {
      try {
         SmartCard.start();

        // Wait for smart card to be inserted
        CardRequest cr = new CardRequest();
        cr.setWaitBehavior(CardRequest.ANYCARD);
        SmartCard sc = SmartCard.waitForCard(cr);

       // Make sure card is available
       if (sc != null) {
         PassThruCardService ptcs = (PassThruCardService)
         sc.getCardService(PassThruCardService.class, true);

         // Build an ISO command to select the MF
         byte[] apdu = { (byte)0x00, (byte)0xA4, (byte)0x00,
         (byte)0x00, (byte)0x02, (byte)0x3F, (byte)0x00 };
         CommandAPDU command = new CommandAPDU(apdu);

        // Send command and get response from card
        ResponseAPDU response = ptcs.sendCommandAPDU(command
            );

        // Print the SW of the response, but a more complex
        // handling could be done
         System.out.println("SW: " +
                                Integer.toHexString(response.sw()))
                                    ;
         // We are done with this smart card.
         sc.close();
       }

     } catch (Exception e) {
       e.printStackTrace();

     } finally { // even in case of an error.
        try{
           SmartCard.shutdown();
        } catch (Exception e) {
        e.printStackTrace();
        }
     }

     System.exit(0);
   }
}
```

Listing 12.1 Configuration for `opencard.properties`

12.3 JSR268: Java Smart Card I/O API

If Java Smart Card I/O API [5] shares some concepts with OCF and PC/SC, it implements communication with smart cards using ISO 7816-4 APDUs, to allow Java applications to interact with the smart cards application. Contrary to OCF, as depicted in Fig. 12.3, it offers a compact architecture but it is sufficient for most applications.

12.3.1 Overview

After two years of discussions inside the Java Community Process, the final version of JSR268 has been released in December 2006. Since the version 6 of Java SE, Java Smart Card I/O API is integrated in the JDK.

The default implementation of the `TerminalFactory` is built on the top of PC/SC, but any other service providers can offer another implementation using the `TerminalFactorySpi` like [7], or even like the SCUBA project [8], which proposed implementations enabling among other features to communicate with CREF (Java Card reference implementation emulator of Oracle) or JCOP emulator (Java

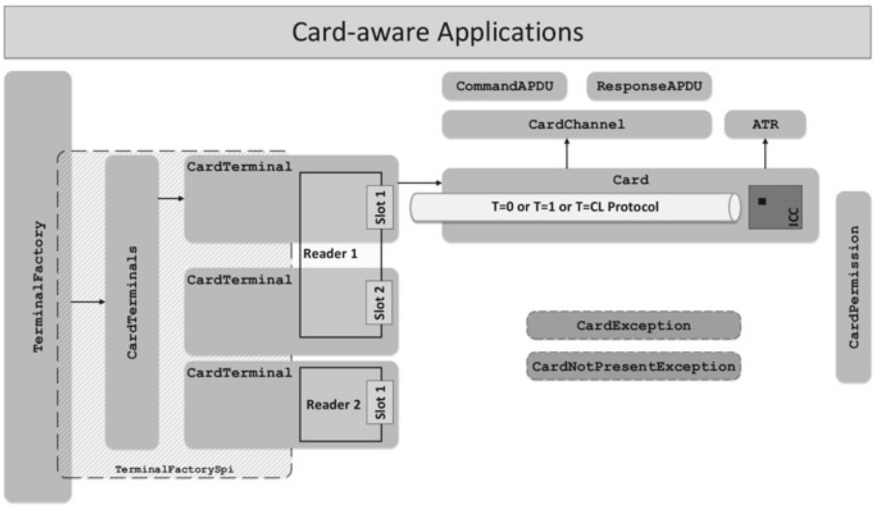


Fig. 12.3 Overview of the JSR268 architecture

Card emulator of NXP). In addition, there also exist at least two re-implementations of the `javax.smartcardio` [7, 9] which contain some workarounds that might interest developers.

In this framework, `TerminalFactory` provides an object of type `Card Terminals` through the `getInstance` method, which reflects the list of the `CardTerminal` available for the targeted implementation (if no parameter is provided to `getInstance`, the default implementation is used, and thus `Card Terminals` is the list of readers available via PC/SC; if a parameter is correctly provided, `CardTerminals` is the list of the readers available through the service provider implementation).

Once the list of readers is obtained, the developer selects a `CardTerminal`, for instance by name or even because the reader contains a card. It must be noted, as illustrated in Fig. 12.3, that in Java Smart Card I/O API, there is one `CardTerminal` per slot of a reader. Note also that there exists the possibility to set permissions for specific actions by providing specific rules including the name of the readers to `CardPermission`. It is mainly interesting for a service provider who wishes to facilitate use of some kinds of cards by providing an abstraction with some "Card-Service" similar to those present in OCF.

When a `CardTerminal` contains a card, the connection can be achieved with `connect` method to obtain an instance on `Card` and by specifying the protocol ("T=0", "T=1", or "T=CL", or "*"). Then the developer has just to build some `CommandAPDU` to transmit to the card with the `transmit` method which provides a `ResponseAPDU` as result.

One additional reason to develop in Java is the possibility to build web Applets that can be loaded in the browser through a web page and that can interact with the readers and thus the cards on the host machine. It is possible to provide a web interface using interactions: of JavaScript and Applet (with the `JSObject`) and of JavaScript and the web page. A developer who masters these technologies can develop applications with end-to-end authentication for instance. This chapter will not say more on this point due to the ever-increasing restrictions on Java applets introduced by all browsers to overcome Java security-related issues. However, if reader of this book wants to know more, a technical guide [10] was written on this topic by SpringCard team.

12.3.2 Example

Hereafter is a code sample selecting the MF of the default selected application.

```
import java.util.List;
import javax.smartcardio.*;

public class SmartCardApplication {
 public static void main(String[] args) {
  try {
   // Display the list of terminals
   TerminalFactory factory = TerminalFactory.getDefault();
   List<CardTerminal> terminals = factory.terminals().list()
       ;
   System.out.println("Terminals:" + terminals);

   // Use the first terminal
   CardTerminal terminal = terminals.get(0);

   // Wait for card to be inserted
   terminal.waitForCardPresent(0);

   // Connect with the card
   Card card = terminal.connect("*");
   System.out.println("card:" + card);
   CardChannel channel = card.getBasicChannel();

   byte[] capdu = {(byte)0x00, (byte)0xA4, (byte)0x00,
   (byte)0x00, (byte)0x02, (byte)0x3F, (byte)0x00};
   ResponseAPDU answer =
             channel.transmit(new CommandAPDU(capdu));
   System.out.println("answer:" + answer.toString());

   // Disconnect the card
   card.disconnect(false);
  } catch(Exception e) {
   System.out.println("Error:" + e.toString());
  }
 }
}
```

Listing 12.2 Code sample selecting the MF of the default application

12.4 PC/SC

PC/SC is a standard of communication between an application on a host and a smart card terminal and consequently between the application and the card contained in the reader.

12.4.1 Overview

In order to standardise an architecture to interface the smart card with a PC, a consortium of companies formed the PC/SC Workgroup [11] in May 1996. It gathered manufacturers of PCs, readers and smart cards industries, including Apple, Bull, Gemplus, Hewlett Packard, Infineon, Intel, Microsoft, Schlumberger and Toshiba.

The PC/SC Workgroup identified 3 parts to be standardised:

- the interface between the reader and the PC (Personal Computer);
- a high-level API to use the functionalities of the card;
- mechanisms allowing several applications to share common resources, such as the card or the reader.

The members of the consortium also wished to define an architecture and specifications allowing cards and reader manufacturers to independently develop their products while ensuring that they can work together.

In December 1997, the working group published the “Interoperability Specification for ICCs and Personal Computer Systems” [12], which is nowadays more commonly known under the name of PC/SC (PC/Smart Card).

Version 1.0 comprised eight parts; each one detailing an aspect of the architecture. Version 2.0 was released in August 2004 followed by an update in January 2006, which added support for contactless cards, enhanced readers (e.g. a keyboard, or a biometric sensor or a screen) and synchronous cards. A second update [13] has been published in November 2013 to enhance the support of readers with secure PIN entry capabilities. It now comprises ten parts which are described in the following section.

12.4.2 Architecture

Figure 12.4 presents what each part of PC/SC 2.01.14 specifications covers:

- part 1 gives an overview of the architecture of the system and components defined by the working group.
- part 2 details the requirements to ensure interoperability between the smart card (i.e. ICC, Integrated Chip Card) and the reader (i.e. IFD, InterFace Device):

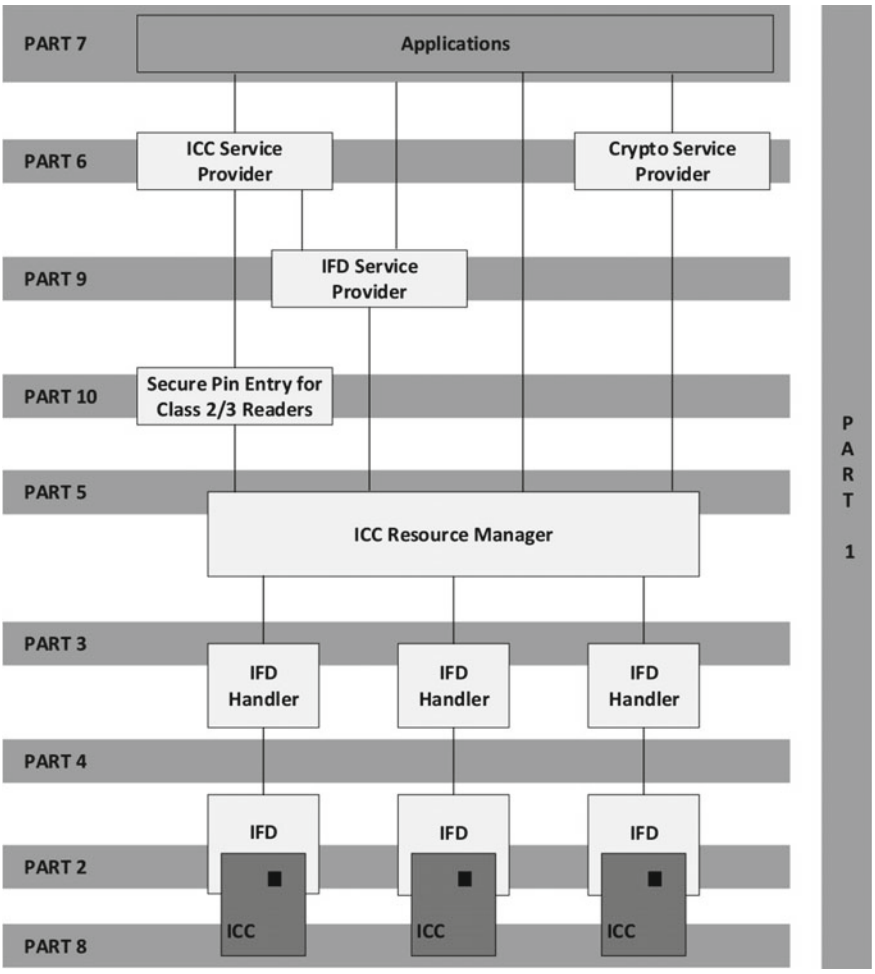


Fig. 12.4 The architecture of PC/SC 2.01.14

  - physical characteristics;
  - communications protocols and the management of errors. They are mainly a copy and paste of the ISO 7816 and ISO 14443 standards in order to manage the contact and/or contactless synchronous and asynchronous cards.

- part 3 describes the interface which the driver of a reader (so-called IFD Handler) must implement so that the higher layers of PC/SC can independently communicate with it, without knowledge of the underlying connector (e.g. USB, serial) and protocols used. In particular, this driver must implement the sending of commands, the powering and unpowering, etc.

- part 4 presents the various types of interface with a reader (e.g. PS2, USB, serial, PCMCIA) and provides a certain number of recommendations which the readers should respect.
- part 5 describes the interfaces and the functionalities supported by the resource manager. It is the central element of the PC/SC system. Indeed, it is responsible to the system resources relating to the cards (i.e. ICC Provider Service) and access to the readers, and thus through them, to the cards.
  In theory, each operating system should provide its own resource manager. Moreover, it solves three problems to manage the access to various readers and cards:
  - it is responsible for the identification and the detection of the resources (e.g. insertion/withdrawal of a card, connection/disconnection of a reader);
  - it is responsible for the allocation of the resources: the cards and the readers for the various applications, by providing mechanisms for sharing or exclusiveness;
  - it proposes a mechanism of transactions to the high layers of PC/SC making it possible to carry out complex operations while ensuring that information of the intermediate states was not corrupted.
- part 6 proposes to hide the complexity of the underlying architecture while proposing to the application programmers a high-level API corresponding to the operations which the card will be able to carry out.
  This part:
  - describes the models for the providers of service in a general way (i.e. ICC Service Provider) and for the providers of cryptographic services (i.e. Crypto Service Provider). Note that the separation of cryptography is related to the restrictions for its import and its export in certain countries;
  - identifies the necessary interfaces;
  - indicates how these models can be extended to meet the needs of applications in specific fields.
- part 7 presents methods for the development of the applications and describes how to use the other components.
- part 8 describes the recommended functionalities for the cards which support cryptography. It contains best practice on the way of managing identification, authentication and protected storage, but also ways of ensuring the integrity of information, its traceability and confidentiality in a solution are based on the card.
- part 9 describes how to manage readers with additional features (i.e. IFD Service Provider): biometric sensor, screen, etc.
- part 10 presents how to deal with secure PIN entry for class 2/3 readers (class 2 readers have a pinpad for secure pin entry whereas class 3 readers have pinpad and a display).

To summarise, in the PC/SC architecture, the card-aware applications are built on the top of one or more Service Providers and the resource manager. The provider of services encapsulates the expected functionalities for a specific smart card and makes them accessible through a high-level API. The Resource Manager handles the resources relating to the cards and the readers inside the system. It allows communication with the readers and through them with the smart cards, via a simple API, on top of which the services of higher levels are built.

12.4.3 Various Implementations

At the beginning, implemented exclusively by Microsoft, a member of the consortium, PC/SC gave access to the card readers using various programming languages but only under Windows. Thus, in 2000, an open source project, `pcsc-lite` [14], initiated by David Corcoran—and in which one of the authors of this chapter, Damien Sauveron, was an active developer—was born to give access to the readers on other operating systems.

12.4.3.1 Microsoft PC/SC

Microsoft, as a member of the working group, was the first to provide an implementation of the PC/SC specification. At the beginning, it consisted of an external program, the resource manager, and a high-level library, the SCard API (for Smart Card) [15] making it possible to communicate with it.

The basic operations allowed by the API are as follows:

- `SCardEstablishContext` This function establishes the resource manager context (the scope) within which database operations are performed.
- `SCardReleaseContext` This function closes an established resource manager context, freeing any resources allocated under that context.
- `SCardListReaders` This function provides the list of readers within a set of named reader groups, eliminating duplicates.
- `SCardConnect` This function establishes a connection, using a specific resource manager context, between the calling application and a smart card contained by a specific reader.
- `SCardDisconnect` This function terminates a connection previously opened between the calling application and a smart card in the target reader.
- `SCardReconnect` This function re-establishes an existing connection between the calling application and a smart card.
- `SCardReleaseContext` This function closes an established resource manager context, freeing any resources allocated under that context.
- `SCardStatus` This function provides the current status of a smart card in a reader. It can be called at any time after a successful call to `SCardConnect` and before a successful call to `SCardDisconnect`.
- `SCardGetStatusChange` This function blocks execution until the current availability of the cards in a specific set of readers changes.

- `SCardTransmit` This function sends a service request to the smart card and expects to receive data back from the card.

More advanced operations are also possible:

- `SCardBeginTransaction` This function starts a transaction, waiting for the completion of all other transactions before it begins.
- `SCardEndTransaction` This function completes a previously declared transaction, enabling other applications to resume interactions with the card.
- `SCardControl` This function gives a direct control of the reader. It can be called at any time after a successful call to `SCardConnect` and before a successful call to `SCardDisconnect`.

This API is an implementation of suggested APIs in part 5 of the PC/SC specifications. For a long time, the first versions of the Resource Manager had many problems. For example, it was only possible to communicate with one reader at a time. After having solved, some of these problems, Microsoft now integrates these components in all of its operating systems.

In this model, for each one of its reader types, each reader's manufacturer has to provide a driver (i.e. IFD Handler) respecting the Microsoft API for the support of the readers (e.g. an API similar to that defined in part 3 of the PC/SC specifications). Indeed, according to part 4 of the specifications, various readers are able to use a specific protocol even if they use the same media (e.g. USB), one driver per reader type or at least by family of reader type using the same protocol for the same media is required. Moreover, drivers need to pass tests of the Windows Hardware Quality Labs [16] in order to be approved.

However, in order to solve these worrying and expensive problems of installation of drivers and certification, an initiative aimed at standardising a protocol on USB was launched within the USB Implementers Forum [1]. Thus, the CCID 1.0 specification (i.e. USB Chip/Smart Card Interface Device) [17] has been developed and released in 2001. An updated version 1.1 [18] has been released in 2005. Microsoft has implemented this specification in a driver distributed with its operating systems thus making it possible for any compliant CCID reader to work without requiring the installation of any additional driver at the time of its first connection of the device. If not yet all readers follow this standard, now most of the manufacturers propose several readers using the CCID standard.

In the PC/SC model, it is still the responsibility of the cards Issuers to provide adequate Service Provider (i.e. a high-level API—part 6 of the PC/SC specifications) for its cards, in order to hide the complexity of the underlying protocol stack. For example, if an Issuer provides a card with a file system, it is probable that it will also provide a Service Provider, making it possible to read and write a file, to create a directory, to move it in the tree structure, etc. For the application programmer who will want to use these cards to store information, it will be more convenient to use this high-level API rather than to communicate directly with the card in a rather obscure language.

If an ICC Service Provider could be related to a particular card of a particular manufacturer, it can also be related to a family of cards compliant with the same standard and thus to cards from various manufacturers. Once again, this example can be illustrated with cards having a file system compliant with the ISO7816-4 standard. If one ICC Service Provider exists for this type of cards then the application programmer will be able to use it in a transparent way without having to be concerned with the particular card manufacturer. Of course, the ICC Service Providers can co-exist without any problem. It should be noted that a clear goal for the working group is to promote adoption of PC/SC via a fair and open standards approach.

Crypto Service Providers (CSP) have a similar role to that of the ICC Service Providers, and they are also standardised in part 6 of the PC/SC specifications. In certain countries, cryptographic algorithms are subject to very strict rules of import and export. Thus, there can be a CSP for a specific card or many general CSPs for a type of card.

To summarise, the ICC Service Provider and Crypto Service Provider are provided in general by the card Issuers or by vendors of interoperable solution components (e.g. file system). It will be noted in addition that these CSPs are at the base of the cryptographic standard of Microsoft: CryptoAPI or CAPI [19].

12.4.3.2 `pcsc-lite`

The project `pcsc-lite` [14] was initiated in 2000 [20] by David Corcoran of the Purdue University. This project was an open source implementation of standard PC/SC 1.0 under BSD licence. The goal of the developers of this project was to provide a better support for the smart cards within the Unix environments. Their efforts were gathered within MUSCLE (Movement for the Use of Smart Card in A Linux Environment) [21] which has disappeared as such, but at the benefit of lot of other smart card projects.

In order to provide a simple API to developers and to facilitate the porting of the card applications developed under Windows, it was decided to implement the SCard API proposed by Microsoft. Nowadays, `pcsc-lite` works on lots of platforms: Linux, Solaris, MAC OS X, FreeBSD, HP-UX, etc. It could be noted that in 2000 Apple has joined the PC/SC working group and has adopted in 2002 `pcsc-lite` as the starting point of its implementation of reference for the Mac OS X systems.

The term "lite" in the name of the project reminds that `pcsc-lite` tries to implement the PC/SC specifications, but it does not include all its functionalities (specially services from part 6 are missing). The most important functions of the Microsoft SCard API are implemented, and it can be checked in reference material [22]. There are still some subtle differences that are mentioned in the API detailed description [23].

Currently, the project includes a resource manager and a partial implementation of the SCard API, which respectively appear as a daemon and a shared library. The `pcsc-lite` daemon is able to support simultaneously a large number of readers and client applications.

The basic operation of the daemon during its initialisation is as follows:

- allocation of the structures of data for the readers and creation of public memory space containing information on the state of the readers (through a shared memory mechanism);
- reading of the configuration file of the static readers (e.g. serial readers);
- creation of the main server and waiting of a client request;
- launching of a thread of detection for the added readers via "hotplug" mechanisms (e.g. USB, PCMCIA).

For each detected reader, if the driver of the reader (i.e. IFD Handler) allows it, the daemon `pcscd` will create a thread to deal with the management of the communication and to give simultaneous access to the resources.

At the time of the first client request (i.e. `SCardEstablishContext`), through the shared library, `libpcsclite` and thus through the Unix sockets, the daemon will create a thread which will start a dedicated server to deal with the context of the client. The subsequent requests will use this new Unix socket until the application decides to finish its dialogue with the daemon (i.e. `SCardReleaseContext`) or it dies (e.g. segmentation problem, termination).

Since 2003, the project `pcsc-lite` has played a major role in the smart cards world, thanks to Ludovic Rousseau who still maintains the project, provides an active support on the mailing-lists and has developed a free implementation of an IFD Handler for the readers respecting the standard CCID [24] under a GPL licence. This driver, which also supports the readers having advanced features, has maintained the level of use of the smart cards on Unix environments at least at the same level as that on Microsoft operating systems.

12.4.3.3 A Few More Features

For the sake of completeness, some useful tools to spy the PC/SC stack exist and deserve to be mentioned. On Windows, there is the PC/SC APDU inspection and manipulation tool [25], and for Unix-like systems, there is the PCSC API spy [26, 27]. Users of these tools have to take care of the data logged since they may contain some keys, PINs or other sensitive data.

12.4.4 Wrappers

An important advantage of PC/SC vs OCF and vs JSR268 in term of popularity is the existence of several *wrappers*. These middleware libraries make it possible to write applications communicating with the resource manager in different programming languages to that provided and imposed by the native library. For example, `pcsc-lite` only provides one library implementing the SCard API for the

C language, and this makes it possible only to write programs in this language or languages able to call C functions.

Therefore, wrappers appeared for Perl [28], Python [29], Prolog [30], Ruby [31], Java [32] and lot of other languages [33]. They use the mechanisms of Foreign Function Interfaces (FFI) [34] of these various languages. These middlewares contribute to the popularity of PC/SC while allowing more programmers to easily access to the low-level functionalities of the PC/SC standard. It is important to note that wrappers also benefit in that a large number of readers are supported by PC/SC thanks to the existing CCID implementations.

12.4.4.1 Two Additional Solutions for Java

In addition of OCF and Java Smart Card I/O API, to access readers and cards with the Java language, there are two solutions based on PC/SC wrappers.

JPC/SC

The first solution is a wrapper called JPC/SC and developed by Marcus Oestreicher of IBM BlueZ Secure Systems. The package supports basically two classes, `Context` and `Card` that are familiar to SCard API programmers, The `Context` and `Card` class methods are mapped, pretty much straight to native functions used in the SCard API, by the use of JNI (Java Native Interfaces) to return the calls of method and the Java arguments towards the native client library provided by the PC/SC Microsoft or by `pcsc-lite`. With `Context`, the programmer establishes the communication with the PC/SC daemon, queries readers and set up a connection with a card. This results in a `Card` instance which is used to exchange APDUs with the card. Errors are typically signalled by `PCSCExceptions`. The `State` class wraps structures such as `READER_STATE` and is used to return the status of readers and card connections such as in `Context.GetStatusChange()` or `Card.Status().`

OCFPCSC Bridge

The other solution is the OCFPCSC bridge [35], i.e. a bridge between the OpenCard Framework and PC/SC. Indeed, even though the OCF project makes it possible to access readers from Java, there were too few OCF drivers (i.e. Card Terminals); using the bridge extends the set of the readers that can be used to all those supported by PC/SC. From the OCF point of view, it is in fact a simple Card Terminal to declare in the configuration file `opencard.properties` and which is connected, through JNI calls, to the native client SCard library of the system to detect the presence of the readers and to communicate with them.

To Conclude on Java solutions

However, in Java, it is always necessary to use a layer of JNI calls to gain access to a hardware level (e.g. `javax.comm` which used JNI is called to implement a `CardTerminal` connected on a serial port in OCF or JSR268 architecture) and to access another native level (e.g. for the use of the solution OCFPCSC bridge), certainly the OCFPCSC bridge solution adds an additional layer in an already complex stack.

The JPC/SC solution has the advantage of being simple and low level compare to OCF and even to JSR268. However, this latter point might be an advantage and a disadvantage. Of course, it gives a greater freedom, but it also requires the knowledge of certain underlying concepts to communicate with a card (APDU, etc.) that is not the case with the OCF CardService.

JSR268 is perhaps the better compromise, since it is less low level than JPC/SC and provides a real framework but less engineering than that of OCF. To be fair, somehow JSR268 includes a PC/SC wrapper in the default implementation of the `TerminalFactory`. Certainly, the best argument in favour of the JSR268 is that it is provided with any JRE.

Anyway, the reader of this book has the choice among different solutions and he/she should know enough now to choose the best for his/her needs.

12.4.5 Examples

One of the best ways to understand how to communicate with a smart card via a reader is to review software source code examples. The following example considers a JPC/SC approach; however, Appendix A of this book also includes C Language and Perl Language examples. The source code is best read with reference to the SCard API.

12.4.5.1 JPC/SC

```
package samples;

import com.linuxnet.jpcsc.*;

public class Test{

   static final int PROTO=PCSC.PROTOCOL_T0|PCSC.PROTOCOL_T1;

   public static void main(String[] args){
      int j = 0;
      if (args.length != 0)
         j = Integer.decode(args[0]).intValue();
```

```
        System.out.println("EstablishContext(): ...");
        Context ctx = new Context();
        ctx.EstablishContext(PCSC.SCOPE_SYSTEM, null, null
            );

        // list readers
        System.out.println("ListReaders(): ...");
        String[] sa = ctx.ListReaders();
        if (sa.length == 0){
            throw new RuntimeException("no reader found");
        }
        for (int i = 0; i < sa.length; i++){
            System.out.println("Reader:" + sa[i]);
        }

       // get status change: wait the insertion of a card
       System.out.println("GetStatusChange() for reader"
       + sa[j] + ": ...");
       State[] rsa = new State[1];
       rsa[0] = new State(sa[j]);
       do{
           ctx.GetStatusChange(1000, rsa);
       }while((rsa[0].dwEventState & PCSC.STATE_PRESENT)
       != PCSC.STATE_PRESENT);
       System.out.println("ReaderState of" +
                                      sa[j] + ":");
       System.out.println(rsa[0]);

      // connect to card
      System.out.println("Connect(): ...");
      Card c = ctx.Connect(sa[j], PCSC.SHARE_EXCLUSIVE,
      PCSC.PROTOCOL_T0|PCSC.PROTOCOL_T1);

      // card status
      System.out.println("CardStatus(): ...");
      State rs = c.Status();
      System.out.println("CardStatus:");
      System.out.println(rs.toString());

     // select APDU for Master File (3F00)
     byte[] capdu = {
         (byte)0x00, (byte) 0xA4, (byte) 0x00,
                                (byte) 0x00, // select
         (byte) 0x02, // length
         (byte) 0x3F, (byte) 0x00 };

     // select MF, different Transmit() methods used
     try{
          System.out.println("Transmit():
                           try to select the MF ...");
          System.out.println("Command:" +
            Apdu.ba2s(capdu, 0, ...... capdu.length));
```

```
                byte[] rapdu = c.Transmit(capdu, 0, capdu.length)
                    ;
                 System.out.println("Response:" +
                   Apdu.ba2s(rapdu, 0, ...... rapdu.length));
            }catch(PCSCException ex){
                  System.out.println("REASON CODE:" +
                                              ex.getReason());
                  System.out.println("TRANSMIT ERROR:" +
                                              ex.toString());
              }

              // disconnect card connection
              System.out.println("Disconnect() from card ...");
              c.Disconnect(PCSC.LEAVE_CARD);
          }

        // release context
        System.out.println("ReleaseContext() ...");
        ctx.ReleaseContext();
    }
}
```

Listing 12.3 Communication with smart card via a reader

12.4.6 Emulation and Remote Communications

Readers of this book might be interested in some applications to access to distant readers and smart cards or even to emulate them. There exist several solutions and some of them are illustrated hereafter. Using the Virtual Smart Card Architecture [36], it is possible to achieve very easily some nice and powerful operations.

12.4.6.1 Card Emulation

For instance, as illustrated Fig. 12.5, Virtual Smart Card, so-called `vicc` can emulate a smart card and make it accessible through PC/SC thanks to a smart card reader driver for `pcsc-lite` and Windows PC/SC, so-called `vpcd`. By default, to communicate

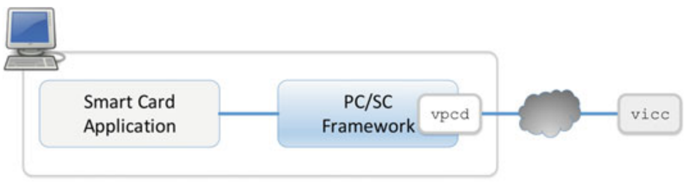


Fig. 12.5 The Virtual Smart Card architecture

with `vicc`, `vpcd` opens slots on local host on port 35963 and port 35964. However, `vicc` does not need to run on the same machine as the `vpcd`. For instance, they can be connected over the internet as will be illustrated in the next section.

It enables users to emulate some cards or even to make bridge to connect external emulators like the already mentioned Java Card emulator of Oracle or the JCOP emulator of NXP. At the time of the writing, the Virtual Smart Card supports at least Generic ISO-7816 smart card including secure messaging, German electronic identity card (nPA), Electronic passport (ePass/Machine Readable Travel Document—MRTD).

12.4.6.2 Access to Remote Reader

As aforementioned by configuring properly `vicc` on a distant host, and `vpcd` on a local host, it is possible to enable a smart card-aware application to access to a remote reader as illustrated in Fig. 12.6.

As illustrated in Fig. 12.7, there is even an Android application, called Remote Smart Card Reader, that enables a host computer to use the smartphones NFC hardware as contactless smart card reader.

12.4.6.3 Other Features

Virtual Smart Card can also be used with PC/SC Relay [37], whose prime purpose is to relay a smart card using a contactless interface, to completely emulate an ISO/IEC 14443 smart card.

The USB CCID Emulator which forwards a locally present PC/SC smart card reader as a standard USB CCID reader can be used as trusted intermediary enabling

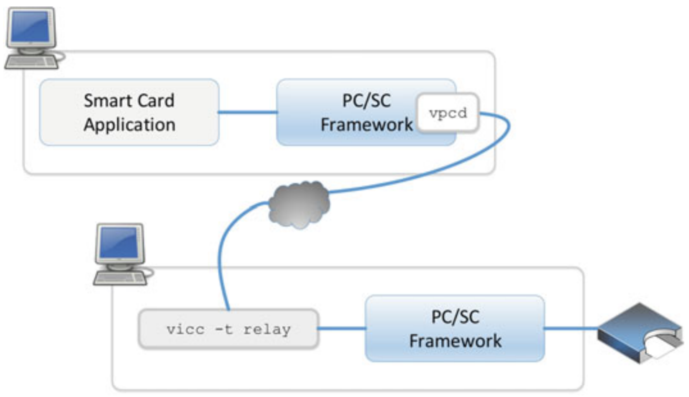
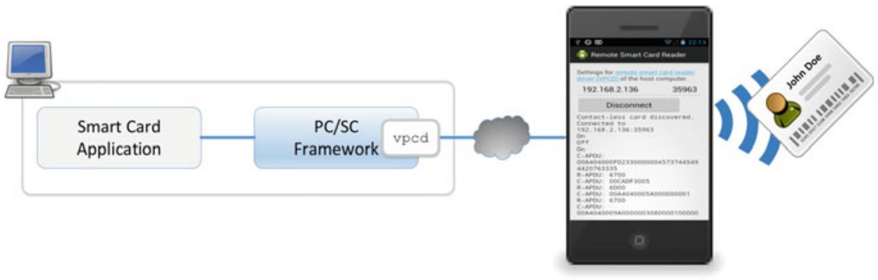


Fig. 12.6 Virtual Smart Card used in relay mode to remotely access a smart card reader and then a card



```
Contact-less card discovered.
Connected to
192.168.2.136:35963
On
Off
On
C-APDU:
00A404000FD2330000004573744549
4420763335
R-APDU:  6700
C-APDU:  00CADF3005
R-APDU:  6D00
C-APDU:  00A4040005A000000001
R-APDU:  6700
C-APDU:
00A4040009A00000030800001000000
```

Fig. 12.7 Mobile phone with NFC used as remote smart card reader to access a contactless card

secure PIN entry and PIN modification. This can be useful to study some parts of a protocol.

Finally, the Android Smart Card Emulator can be mentioned. It allows the emulation of a contactless smart card. It uses Android's HCE (Host Card Emulation) to fetch APDUs from a contactless reader and delegate them to applications. These applications can be Java Card Applets running on an embedded version of JCardSim (an open source Java Card Runtime Environment Simulator) or applications running elsewhere and connected with `vpcd` running on the mobile Phone.

12.5 Mobile APIs

The aim of this section is to provide the first samples of code that the reader of this book can use to write his/her own applications on a mobile phone to interact with smart cards. However, since there are multiple operating systems for mobile phones such as Android, BlackBerry OS, iOS, Windows Phone to cite a few, an exhaustive presentation is not possible. Moreover, the access to smart card and similar devices (SE, etc.) is not well documented on all the operating systems (for instance, at the time of the writing,[1] Apple has still not released the APIs to access the NFC controllers present of most of the iPhones—certainly to maintain its Apple Pay application as the only one purse, instead to have to compete with Google Wallet, etc.). Consequently, only the Android API is explained. JSR177 is also presented, since it can be used on phones supporting Java (J2ME) to access Universal Integrated Circuit Card (UICC) and Secure Element (SE). JSR257, which is the equivalent of JSR177 for NFC, is not presented. The PhoneGap-NFC library of PhoneGap development framework is not presented, but it might interest the reader of this book since it supports Windows Phone 8, BlackBerry 7, and BlackBerry 10 (but with some limitations). However, to also provide access to smart cards to people owning mobile phone running other operating systems than Android, this section presents proprietary APIs for external smart card readers that can be connected to mobile phone in various ways.

[1]May 2016.

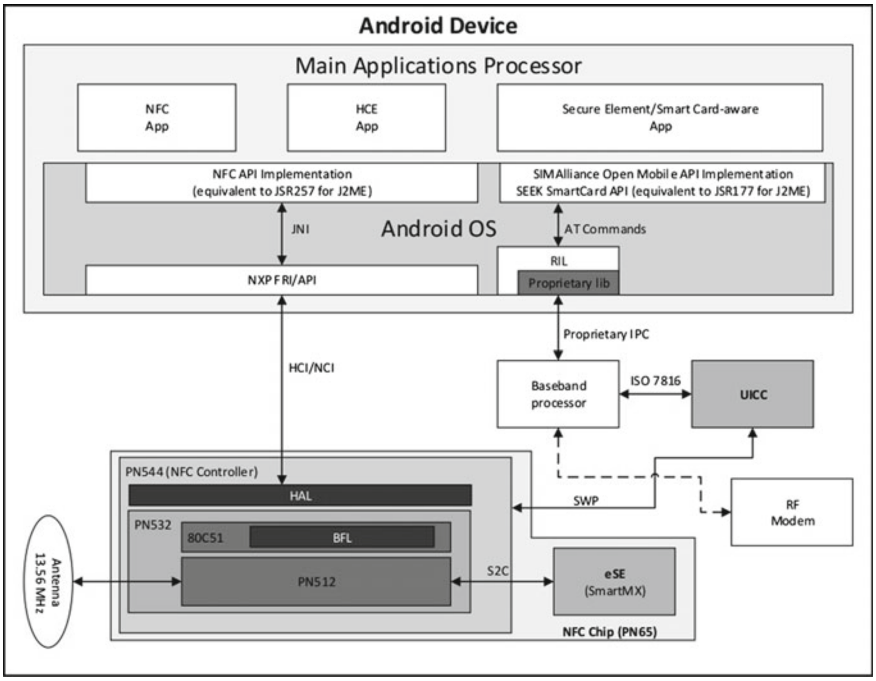


Fig. 12.8 Android architecture for NFC and eSE & UICC (illustrated with Nexus S)

12.5.1 Android APIs to UICC, SE and NFC

Android is the best documented operating system to access NFC elements, and it is quite easy to write applications and to make them operate on a device. Accessing the UICC or the SE is less straightforward, and it may not work on some devices. A workaround against this issue will be to develop applications using an emulator. Figure 12.8 illustrates the internal architecture to access NFC, the SE and the UICC with Android. Note that it is considered here that the NFC controller is provided by NXP (which is certainly the biggest manufacturers of NFC ICs).

12.5.1.1 NFC

Since version 2.3, NFC is supported in Android. HCE (Host Card Emulation) was introduced since version 4.4. In this section, we only present how Android applications can interact with NFC tags, which are basically smart cards, sometimes under different form factors. Thus among the different modes of operation of NFC, only the reader/writer is presented, not the peer-to-peer and card emulation modes.

To access the NFC system service, NfcService, the applications need to use APIs located in the package com.android.nfc. Instead of exposing low-level functionality of NfcService in APIs, an event-driven framework similar to a lot of other services in Android has been proposed; i.e. applications do not have to manage tag discovery. since it is managed by the NfcService which will notify registered applications for the requested events.

The first operation requested to have access to NFC tags in an application is thus to register an Android activity to the tag dispatch system for the requested intents. On tag discovery, the tag dispatch system uses three intents actions to notify the registered activities. Intents are listed in order of highest to lowest priority: ACTION_NDEF_DISCOVERED, ACTION_TECH_DISCOVERED and ACTION_TAG_DISCOVERED. ACTION_NDEF_DISCOVERED is used to start an activity when a tag that contains an NFC Data Exchange Format (NDEF) payload is scanned and is of a recognised type. ACTION_TECH_DISCOVERED is used when the tag does not contain NDEF payload or when it contains an NDEF payload that was not handled by activities registered for ACTION_NDEF_DISCOVERED intent since data cannot be mapped to a Multipurpose Internet Mail Extension (MIME) type or Uniform Resource Identifier (URI) of interest for them. ACTION_TAG_DISCOVERED is used if no activities has handled the ACTION_NDEF_DISCOVERED or ACTION_TECH_DISCOVERED intents.

Thus, the application must request permission to access the NFC hardware, and it must register the requested NFC events to handle by the related NFC-enabled activity in the AndroidManifest.xml file using the standard intent filter system.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android=
                    "\url{http://schemas.android.com/apk/res/
                        android}"
    package="example" ... >
    ...
    <uses-permission android:name="android.permission.NFC" /
        >
    <uses-feature android:name="android.hardware.nfc"
                                    android:required="true" /
                                        >
    ...
    <application ... >
        <activity android:name=".NFCActivity" ... >
            <intent-filter>
                <action android:name=
                        "android.nfc.action.NDEF_DISCOVERED" /
                            >
                <category android:name=
                            "android.intent.category.DEFAULT" /
                                >
                <data android:mimeType=
                                                    "text/plain" /
                                                        >
            </intent-filter>
            <intent-filter>
                <action android:name=
                        "android.nfc.action.TECH_DISCOVERED" />
            </intent-filter>

            <meta-data android:name=
                          "android.nfc.action.TECH_DISCOVERED"
                android:resource="@xml/nfc_tech_filter" >
            </meta-data>
        </activity>
    ...
    </application>
</manifest>
```

Listing 12.4 AndroidManifest.xml

First, the application requests the permission to access the NFC controller and then declares it uses the NFC features so that the application will be shown up in Google Play only for devices that have NFC hardware. Then, `NFCActivity` is declared to be the activity that will handle the NFC intents for which it is registered; i.e. with `NDEF_DISCOVERED`, it can handle tags with NDEF data with the `text/plain` MIME type, and with `TECH_DISCOVERED`, it can handle tags using the technologies specified in the specified XML resource file, `res/xml/nfc_tech_filter.xml`. The activity can thus handle tags supporting technology ISO14443-4 (`IsoDep`) or tags supporting technologies NDEF (`Ndef`) and ISO14443-3B, often shorted with ISO14443B, (`NfcB`). It means that a NDEF, ISO14443-3B tag with no `text/plain` MIME type will not be handled with the `ACTION_NDEF_DISCOVERED` intent but with the `ACTION_TECH_DISCOVERED` intent.

```
<?xml version="1.0" encoding="utf-8"?>
<resources xmlns:xliff=
    "urn:oasis:names:tc:xliff:document:1.2">
    <tech-list>
        <tech>android.nfc.tech.IsoDep</tech>
    </tech-list>

    <tech-list>
        <tech>android.nfc.tech.NfcB</tech>
        <tech>android.nfc.tech.Ndef</tech>
    </tech-list>
</resources>
```

Listing 12.5 nfc_tech_filter.xml

If the discovered tag does not match any of the declared intents, the tag dispatch system will pass it to another activity or will destroy it if no activity is registered for handling it. The reader of this book might wonder how cases are managed where several activities are registered for the same intents. Android simply shows a selection dialogue to allow the user to select which activity should handle it. However, it can be noted that in `AndroidManifest.xml` file, the `DEFAULT` category has been assigned to `NDEF_DISCOVERED` which means that the `NFCActivity` is a candidate to be the default application to handle the specified type of tags if no other activities request to be the default application; else, a selection dialogue is also shown. It can also be mentioned the possibility for an application in foreground to call the `NfcAdapter.enableForegroundDispatch()` method to short-circuit the selection dialogue and get the highest priority to receive the NFC intents when the application is in the foreground. Note that for our objective which is to be able to select a master file (3F00), `ACTION_NDEF_DISCOVERED` intent for the `IsoDep` technology would have been sufficient.

When an intent is received, the activity has just to handle it. The activity has just to use the `EXTRA_TAG` extra of `NfcAdapter` to get the `Tag` object. Using it, the activity can get the `IsoDep` object to send and receive APDUs. It can be mentioned that the `EXTRA_NDEF_MESSAGES` extra for NDEF tags to access an array of `NdefMessage` present on the discovered tag.

```
protected void onNewIntent (Intent intent) {
 setIntent(intent);

 Tag tag = intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
 IsoDep iso = IsoDep.get(tag);
 if (iso != null) {
 iso.connect();
 byte[] capdu = { (byte)0x00, (byte)0xA4, (byte)0x00,
            (byte)0x00, (byte)0x02, (byte)0x3F, (byte)0x00 };
 byte[] rapdu = iso.transceive(capdu);
 ...
 }
}
```

Listing 12.6 A sample code to get tag object

It can also be mentioned since Android 4.4, a new mode, called reader mode, is also possible to allow an activity to obtain `Tag` object directly when the system finds a tag while this activity is in foreground instead of receiving an intent. In this mode, only the reader/writer mode is available (not the peer-to-peer and card emulation modes). The activities can switch the reader mode on or off when they are in foreground by calling, respectively, the `enableReaderMode()` or `disableReaderMode()` methods of the `NfcAdapter` class. Since in our example, the interest is to get an `IsoDep` object, in `enableReaderMode()` all the types of NFC tags are requested to be polled.

```
public class NFCActivity extends Activity implements
    NfcAdapter.ReaderCallback {private NfcAdapter adapter;
     ...
    @Override
    public void onResume() {
         super.onResume();
         if (adapter != null) {
              adapter.enableReaderMode(this, this,
                NfcAdapter.FLAG_READER_NFC_A |
                NfcAdapter.FLAG_READER_NFC_B |
                NfcAdapter.FLAG_READER_NFC_BARCODE |
                NfcAdapter.FLAG_READER_NFC_F |
                NfcAdapter.FLAG_READER_NFC_V |
                NfcAdapter.FLAG_READER_NO_PLATFORM_SOUNDS |
                NfcAdapter.FLAG_READER_SKIP_NDEF_CHECK, null);
        }
    }

    @Override
    public void onTagDiscovered(Tag tag) {
        IsoDep iso = IsoDep.get(tag);
        if (iso != null) {
        iso.connect();
        byte[] capdu = { (byte)0x00, (byte)0xA4, (byte)0x00,
            (byte)0x00, (byte)0x02, (byte)0x3F, (byte)0x00 };
        byte[] rapdu = iso.transceive(capdu);
        ...
        }
    }
    ...
}
```

Listing 12.7 NFCActivity class to poll NFC tags

12.5.1.2 UICC and SE

As illustrated Fig. 12.8, the internal architecture to access the SE, UICC are quite complex. As mentioned in the caption of the figure, the Nexus S has been used to show the components of the NFC chip for a better understanding. On the Nexus S, the NFC chip is a PN65 which contains different hardware and software components, such as:

- A PN512 NFC transmission module for contactless communication at 13.56 MHz.
- A microcontroller (80C51 core with 32 kB of ROM and 1 kB of RAM) running the firmware for the PN512 transmission module. The combination of the micro-controller and the PN512 is also called PN532. This is a basic NFC controller, so-called on NXP website a highly integrated transceiver module.
- An additional interface and software stack to use UICC as the Secure Element. Therefore, the PN544 chip needs to implement the so-called SWP (Single Wire Protocol). From the Android OS perspective, the PN544 is the real NFC controller and the communications take place between the NXP NFC software stack implementation, so-called NXP FRI (Forum Reference Implementation), and the PN544 using HCI (Host Controller Interface standardised by ETSI) protocol or for more recent NFC controller the NCI (NFC Controller Interface standardised by NFC Forum) protocol. The NXP FRI is implemented in plain ANSI C, and it is called by Java-implemented NFC services and APIs of Android platform through JNI.

- A secure smart card chip which can be used as the embedded Secure Element (eSE). This secure element is connected to the PN531 NFC controller with S2C (SignalIn/SignalOut Connection) which was standardised as NFC-WI—NFC Wired Interface. In this case this is the P5CN072, a Secure Dual Interface PKI Smart Card Controller of the SmartMX platform featuring 160 kB of user ROM, 4608 bytes of RAM and 72 kB of EEPROM which is running Java Card.

Note that HAL and BFL, respectively, stand for Hardware Abstraction Layer and Basic Functions Library. Both software components can be requested through NXP FAE (Field Application Engineer) by qualified customers. This description should have helped the reader of this book to understand the different NFC chips and why, for instance, a PN532 is enough to communicate for an Arduino, or Raspberry Pi, NFC shield, but this is a bit far off the topic of this chapter.

It can be noted that, most of the time, the UICC is also a Java Card which usually runs at least the SIM application (it can be USIM or even CSIM (CDMA SIM), IP Multimedia Services Identity Module (ISIM) or several of them). The UICC can even be based on SmartMX like the eSE of the PN65 which is quite interesting. Since the UICC is connected to NFC controller via the SWP link, in card emulation mode, it can receive the external commands of an NFC reader according the internal configuration, but this is out of the scope of this chapter.

Basically, there are two ways to access the mentioned secure elements: eSE when present and UICC. The first one is to do that through the NFC controller, since when the eSE is present, it is a part of the NFC chip and second via the SWP link since UICC is connected to NFC controller. For this solution, the use of an optional library called `nfc_extras` is required. However, this solution is not presented here, since it is quite complex to set up in practice; readers of this book are advised to read the interesting book [38] and blog [39, 40] of Nikolay Elenkov to get more details. The second solution is to use the SIMAlliance Open Mobile API [41].

Before explaining how to use this API, the reader should know that network authentication is implemented by the baseband processor, and thus, it is never directly visible from the main Android OS. In addition, though Android supports, SIM Toolkit (STK) applications to display menus, send SMS, etc., and though it can look up and store contact in the SIM file system, the Android security overview explicitly states that low level access to the SIM card is not available to third-party apps. However, this might be disappointing, fortunately some manufacturers, like Samsung, provide an implementation of the SIMAlliance Open mobile API in some builds of their smartphones. There is also an open source implementation, supported by several compatible devices, provided by the SEEK (Secure Element Evaluation Kit) Android project [42]. The SIMAlliance Open Mobile API aims to provide an unified interface to access SEs (eSE, UICC, micro SD card—so-called ASSD, i.e. Advanced Security SD—etc.). However as illustrated Fig. 12.8, the Open Mobile API implementation has to communicate with the RIL (Radio Interface Layer), through the telephony framework (which is not presented on the figure) using AT commands (`AT+CSIM` for generic SIM access, `AT+CCHO` for open logical channel, `AT+CGLA` for generic UICC logical channel access, and `AT+CCHC` for close logical channel) defined by

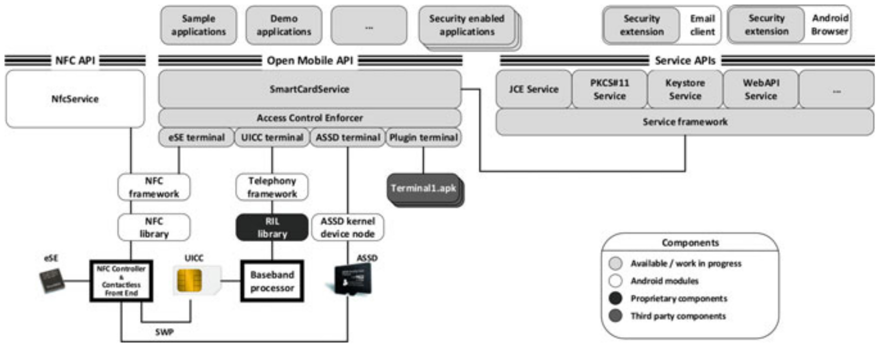


Fig. 12.9 Architecture of the SEEK for Android

3GPP TS 27.007 specification. But the RIL daemon can only provide the baseband processor with the appropriate IPC (Inter-Processor Communication) requests to access the UICC if the proprietary RIL HAL library wrapping these AT commands is provided by the handset manufacturer. In addition, it is important to mention that some manufacturers also use proprietary extensions to implement APDU exchanges instead of `AT+CGLA`; in such a case, it will not be not possible to access the UICC through AT commands and modification of the UICC support in SEEK Open Mobile API will be required to call the appropriate proprietary functions instead of the AT commands—for instance, see modification of RIL for Galaxy S3 to use SEEK [43].

As illustrated Fig. 12.9 SEEK for the Android Platform provides implementation of the `SmartCardService` resource manager service which is able to manage several kinds of SE (eSE, ASSD, UICC, etc.), an extension of the Android telephony framework to enable a transparent APDU exchange with the UICC based on AT commands. However, to be used in real-life, a compatible device and a build that includes the `SmartCardService` and related framework extension are required (since the access through the RIL is hardware-dependent and HAL-dependent—for instance see [43]). Most of recent[2] Samsung Galaxy devices are fine. If needed, the reader of this book can compile his/her own build for his/her device. An alternative that is suggested by the authors of this chapter for the development purpose is to use the UICC support in Android emulator which enables the support of PC/SC reader and then the access to a real UICC card. In this case, developer can develop his/her own Android applications run by Android emulator to access to his/her own Java Card application if the UICC is a Java Card. Detailed explanations to compile the emulator with UICC support or ASSD support are given on the website of the SEEK project [44]. A last point to note is that current version of SEEK implementation, i.e. 4.0.0, is based on Android 5.0.0 (Lollipop), and it supports the SIMAlliance Open Mobile API v3.0 [45].

To be able to use the Open Mobile API, applications must request `SMARTCARD` permission and add the extension library in their `AndroidManifest.xml` file.

[2]May 2016.

```
<manifest ...>
...
    <uses-permission android:name=
               "org.simalliance.openmobileapi.SMARTCARD" />
    <application ...>
    <uses-library android:name=
                               "org.simalliance.openmobileapi"
                      android:required="true" />
    ...
    </application>
...
</manifest>
```

Listing 12.8 AndroidManifest.xml

Then to access the available secure elements, the application first creates an `SEService` object, which connects to the `SmartCardService` asynchronously and notifies the application via the `serviceConnected()` method of the implemented `SEService.CallBack` interface when the connection is established. When notified, the application can then get a list of the available SE readers using the `getReaders()` method. Usually, there is at least one reader, the built-in UICC reader in the device, but there may be more if an eSE and/or an ASSD are present and the `SmartCardService` has been configured to use them. The application can then open a session to the target SE using the `openSession()` method (in our code, the first reader/SE; i.e. `0`). When the session is opened (i.e. the `Session` object is created), the application calls the `openLogicalChannel()` method in order to obtain a `Channel` object, which it then uses to send and receive APDUs using its `transmit()` method.

```
import org.simalliance.openmobileapi.*;


public class OMAActivity extends Activity implements
                                        SEService.CallBack {

    private SEService seService;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        ...
        seService = new SEService(this, this);
        ...
    }

    protected void onDestroy() {
        if (seService != null && seService.isConnected()) {
            seService.shutdown();
        }
        ...
    }

    public void serviceConnected(SEService service) {
        Reader[] readers = seService.getReaders();
        Session session = readers[0].openSession();
        Channel channel = session.openLogicalChannel(null);
        byte[] capdu = { (byte)0x00, (byte)0xA4, (byte)0x00,
           (byte)0x00, (byte)0x02, (byte)0x3F, (byte)0x00 };
        byte[] rapdu = channel.transmit(cadpu);
        ...
        channel.close();
        session.close();
        ...
    }
    ...

}
```

Listing 12.9 Accessing secure element

When using the Android emulator with UICC support, the UICC must contain a SIM application else the emulated phone will be locked. If a SIM application is present, the developer has to know the right PIN code, and he should take care to not lock the SIM card. He should not submit the PIN code more than two times if the emulated phone does not unlock itself, and he should set back the tries at the maximum by presenting the PIN code to the SIM on a real device. It can be noted that the developer who wants to work with a common Java Card that does not have SIM application can write an applet to emulate it. The authors of this chapter have written a minimal applet that should be submitted to the SEEK project in mid-2016. Finally, it should be noticed that if the developer wants to work with a classic Java Card, it will require that the card support at least two logical channels. The first one is used by the emulated handset to manage the connection with the SIM/USIM application. The second one will be used to select the desired Java Card applet.

12.5.2 JSR177

The Security And trust Services API (SATSA, referred to as JSR177 [46]) extends the security features of the J2ME (Java 2 Micro Edition) platforms. In particular, these API proposes to manage a Security Element (SE) that can perform secure operations (storage, computation, etc.), cryptographic operations, etc.

Among the four optional packages of the APIs of this specification, two are related to the smart card communication:

- SATSA-APDU defines an API to support communication with smart card applications using APDUs (`javax.microedition.io.APDUConnection`).
- SATSA-JCRMI defines a Java Card Remote Method Invocation (RMI) client API that allows a J2ME application to invoke a method of a remote Java Card object.

Since SATSA is based on the Connected Limited Device Configuration (CLDC) 1.0 Generic Connection Framework (GCF), developing MIDlets that make use of the smart card on the cellphone is not only relatively easy, but also familiar to J2ME developers.

It can be noted that SATSA brings the J2ME and Java Card platforms closer together. Indeed SATSA caters to both of the overall programming models for Java Card applications: the APDU-message model and the Java Card RMI object-oriented distributed model. For each of these models, SATSA defines a new GCF connection type:

- `APDUConnection` allows a J2ME application to exchange APDUs with a smart card application using the ISO-7816 APDU protocol.
- `JavaCardRMIConnection` allows a J2ME application to use Java Card RMI to invoke remote methods on a smart card.

All GCF connections are created using the `Connector.open()` method. One of the arguments to `Connector.open()` is a URL that indicates the type of connections to create. The CLDC GCF defines this URL as a string in the following format:

```
scheme:[target][params]
```

where:

- `scheme` is the connection type to create (and protocol to use).
- `target` is typically some kind of network address.
- `params` are optional parameters in the form `name=value`, separated by a semi-colon.

For SATSA, the format of the URL is:

```
protocol:[slotID];AID
```

where:

- `protocol` is either `apdu` for an APDU-based connection or `jcrmi` for a JCRMI-based connection.
- `slotID` is the number that indicates the slot into which the card is inserted. The slotID field is optional; default value is `0`.
- `AID` is the application identifier for a smart card application. For example, "`A0.00.00.00.09.00.01`" is the GSM application AID.

12.5.2.1 Example

An `APDUConnection` defines the methods that allow to communicate to ISO7816 compliant cards using GCF. It defines three methods:

- `enterPIN()` prompts the user for a PIN.
- `exchangeAPDU()` exchanges an APDU with a smart card application. This call blocks until a response has been received from the smart card (or processing is interrupted).
- `getATR()` returns the Answer To Reset (ATR) message sent by the smart card in response to the reset operation.

The following code sample shows how to open an APDUConnection, how to close it, and how to exchange a command APDU and receive a response APDU.

```
...
byte[] commandAPDU =
       { (byte)0x00, (byte)0xA4, (byte)0x00, (byte)0x00,
         (byte)0x02, (byte)0x3F, (byte)0x00 };
try {
  // Create an APDUConnection
  String url ="apdu:0;AID=A0.00.00.00.09.00.01";
  APDUConnection ac = (APDUConnection) Connector.open(url);

    // Send a command APDU and receive a response APDU
    byte[] responseAPDU = ac.exchangeAPDU(commandAPDU);
    ...

    // Close connection.
    ac.close();
} catch(IOException e){
    ...
}
...
```

Listing 12.10 Code sample showing how to open an APDUConnection

As shown above, SATSA makes APDU communication relatively simple. More examples can be found in [47].

12.5.3 Proprietary APIs

In this section, the APIs from Feitian are mainly presented since this company proposes several solutions to enable access to smart cards from different mobile operating systems (Android/Blackberry/iOS). However, some products from competitors (like Advanced Card Systems Ltd—ACS) are also presented for fairness, and the reader of this book will choose those which fulfill his/her needs.

12.5.3.1 Bluetooth Readers

Bluetooth readers presented in Fig. 12.10 are interesting products, since usually Bluetooth is supported by most mobile phones. In addition, the two presented products

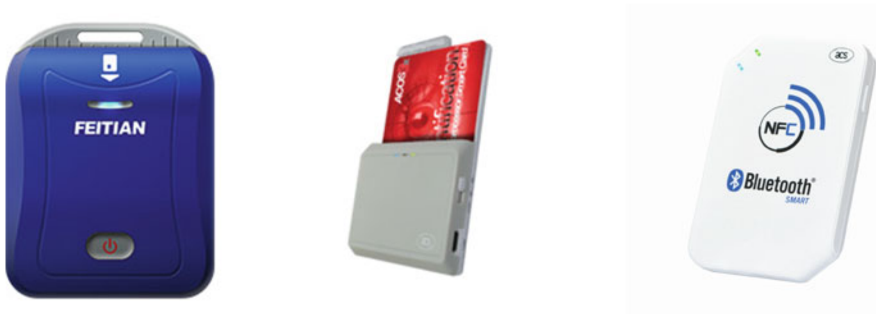


Fig. 12.10 bR301 (Feitian) and ACR3901U-S1 (ACS) for contact card and ACR1255U-J1 (ACS) for contactless card

offer the possibility to communicate with a distant card since the readers are standalone battery-powered, and they can be placed away from the mobile phone to which they are connected. The presented readers also deserve interest since they are CCID-compliant. Thus, if they are plugged via their micro USB port to a host supporting PC/SC and having a CCID driver, applications can access them. There even exists a driver for a Bluetooth connection to the bR301 under Windows. Note that the micro USB port is also used to charge the battery.

Some snippets of code required to use the bR301 on Android are available hereafter. APIs are available here [48], and the full code of samples are available here [49].

```
import java.util.UUID;

import com.feitian.readerdk.Tool.DK;

// here are some members
private ft_reader mReader;
                // ft_reader is provided in the sample
                // It encapsulate the low level communication

// The well known SPP (Serial Port Profile) UUID in Android
private static final UUID MY_UUID =
   UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");

BluetoothAdapter mBlueToothAdapter = null;
ArrayList<BluetoothDevice> arrayForBlueToothDevice;
BluetoothDevice mBlueToothDevice;
BluetoothSocket mBlueToothSocket;
InputStream mInput;
OutputStream mOutput;

...

// To get the BlueTooth adapter — Usually this code is in
// public void onCreate(Bundle savedInstanceState)

```

```
mBlueToothAdapter = BluetoothAdapter.getDefaultAdapter();
if  (mBlueToothAdapter == null) {
    Toast.makeText(this, "this device does not support
    bluetooth",Toast.LENGTH_LONG).show();
    finish();
    return;
}
// Create an object to maintain the list of Bluetooth
                                                    devices
arrayForBlueToothDevice =
                        new ArrayList<BluetoothDevice>();

...

// To build the list of Bluetooth devices with
                                          "FT" inside the name
arrayForBlueToothDevice.clear();
Set<BluetoothDevice> pairedDevices=
                      mBlueToothAdapter.getBondedDevices();
if (pairedDevices.size() > 0) {
    for (BluetoothDevice device : pairedDevices) {
        String str = device.getName();
        if (null != str && (-1 != str.indexOf("FT")))
           arrayForBlueToothDevice.add(device);
...

// To select a device in the list —
                                     Let say the device 0 here
mBlueToothDevice = arrayForBlueToothDevice.get(0);

...

// To create the socket to the reader
mBlueToothSocket =
mBlueToothDevice.
   createInsecureRfcommSocketToServiceRecord(MY_UUID);
mBlueToothSocket.connect();
mInput = mBlueToothSocket.getInputStream();
mOutput = mBlueToothSocket.getOutputStream();

// To create the reader object
mReader = new ft_reader(mInput, mOutput);
displayData("System" , "create socket ok");

// To register a handler
mReader.registerCardStatusMonitoring(mHandler);

...
private final Handler mHandler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        switch (msg.what) {
        case DK.CARD_STATUS:
                switch (msg.arg1) {
                case DK.CARD_ABSENT:
                    ...
                    break;
                case DK.CARD_PRESENT:
                    //Power On
                    mReader.PowerOn();
                  byte[] cAPDU = { (byte)0x00, (byte)0xA4,
                  (byte)0x00, (byte)0x00, (byte)0x02,
                              (byte)0x3F, (byte)0x00 };

                    byte[] rADPU = new byte[1024];
                    int[] length = new int[2];
                    int ret = DK.RETURN_SUCCESS;
                    try {
                        ret = mReader.transApdu
                    (cAPDU.length, cAPDU, length, rAPDU);
                    } catch (Exception e) {
                        ...
                    }
                    break;
                case DK.CARD_UNKNOWN:
                        ...
                        break;
                case DK.IFD_COMMUNICATION_ERROR:
                        ...
                        break;
                }
            ...
            default:
                    break;
            }
    }
}; ...

// To close device connection
mReader.PowerOff();
mReader.readerClose();
mBlueToothSocket.close();
mBlueToothSocket = null;
```

Listing 12.11 Code snippets to use the bR301 on Android

Using the bR301 on iOS is very simple since Feitian provides an API [50] similar to SCard. This API is also usable with the iR301 presented in Sect. 12.5.3.4. Samples of code in Objective C and Swift are available here [51].

12.5.3.2 Audio Jack Readers

For mobile phones without built-in support of smart cards, there exist readers that can communicate through the Audio Jack connector of the mobile phone. Usually

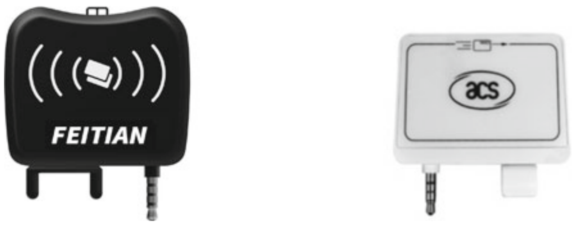


Fig. 12.11 aR530 (Feitian) for contactless card and ACR32 (ACS) for contact card

these readers are also battery-powered and can be charged via a micro USB port. Figure 12.11 presents the aR530 from Feitian which is a reader for contactless smart card, and the ACR32 from ACS which is a reader for contact smart card. Note that a previous version of this aR530 included a reader for magnetic stripe. The presented ACR32 includes a magnetic stripe reader. Feitian also proposed the aR301 for contact card, and ACS proposed the ACR35 for contactless card with support of magnetic stripe reading.

Some snippets of code required to use the bR530 on Android are available hereafter. APIs are available here [52], and the full code of samples are available here [53].

```
import com.ft.mobile.reader.Card;
import com.ft.mobile.reader.Convert;


// here are some members
public static Card myCard;

// To create the connection
public void onCreate(Bundle savedInstanceState){
    ...
    myCard = new Card();
    myCard.Initial(this);
    ...

...

// Wait for a card (to do in an AsyncTask)
ArrayList<Card.CARD_TYPE> list =
                    new ArrayList<Card.CARD_TYPE>();

list.add(Card.CARD_TYPE.A_CARD);
list.add(Card.CARD_TYPE.B_CARD);
list.add(Card.CARD_TYPE.Felica_CARD);
list.add(Card.CARD_TYPE.Topaz_CARD);
```

```
Card.CARD_TYPE[] types =
                    new Card.CARD_TYPE[list.size()];
list.toArray(types);
int rc = myCard.FTNFC_connect(types, 5);
return rc;

...

// To send the APDU command
byte[] sendData =
            Convert.hexStringTo16("00A40000023F00");
byte[] respData = new byte[1024];
int recvLen =
       myCard.FTNFC_transmitCmd(sendData, respData);

...

// For the disconnection
int rc = myCard.FTNFC_disconnect();
```

Samples of code in Objective C for iOS are available here [54]. The API is here [55].

12.5.3.3 Micro USB OTG Readers

Another solution is to use the micro USB OTG (On-The-Go) port of the mobile phone when available. ACS and Feitian propose products presented in Fig. 12.12.

If the reader of this book is interested to know more, he/she is invited to check on the website of the manufacturers.

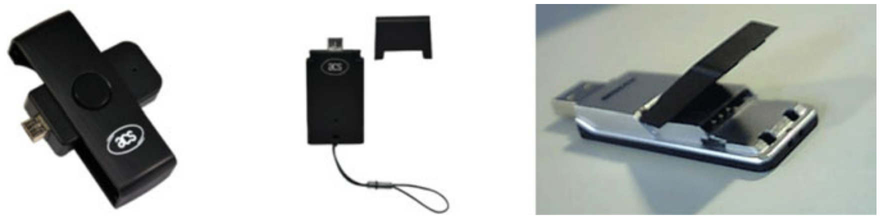


Fig. 12.12 ACR38U (ACS) for full-sized contact card, ACR39T-A3 (ACS) and R301 B5 Casing (Feitian) for SIM-sized contact card

Fig. 12.13 Two connectors-different iR301 (Feitian) for contact card with and an iPad Air casing reader (Feitian) for contact card

12.5.3.4 Proprietary Port-Connected Readers

Feitian also proposes products for proprietary connectors like those existing on Apple mobile phones and tablets to enable these devices to access smart cards. Some of the products are presented in Fig. 12.13.

As with the bR301, samples of code for these smart card readers and API are available here [50, 51].

12.6 In Conclusion

This chapter has discussed the main middleware that can be used to access smart cards via card readers, plus the associated standards bodies and workgroups both on a host and a mobile phone. Now, the reader of this book is invited to play with the APIs.

Appendix A

C Language

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>
#include <string.h>

#include <PCSC/wintypes.h>
```

```c
#include <PCSC/winscard.h>

#ifndef TRUE
#define TRUE 1
#define FALSE 0
#endif

/* PCSC error message pretty print */
#define PCSC_ERROR(rv, text) \
if (rv != SCARD_S_SUCCESS) \
{ \
    printf(text ": %s (0x%lX)\n", pcsc_stringify_error(rv), rv); \
    goto end; \
} \
else \
{ \
    printf(text ": OK\n\n"); \
}

int main(int argc, char *argv[])
{
    LONG rv;
    SCARDCONTEXT hContext;
    DWORD dwReaders;
    LPSTR mszReaders = NULL;
    char *ptr, **readers = NULL;
    int nbReaders;
    SCARDHANDLE hCard;
    DWORD dwActiveProtocol, dwReaderLen, dwState, dwProt, dwAtrLen;
    BYTE pbAtr[MAX_ATR_SIZE] = "";
    char pbReader[MAX_READERNAME] = "";
    int reader_nb;
    unsigned int i;
    const SCARD_IO_REQUEST *pioSendPci;
    SCARD_IO_REQUEST pioRecvPci;
    BYTE pbRecvBuffer[10];
    BYTE pbSendBuffer[] = { 0x00, 0xA4, 0x00, 0x00, 0x02, 0x3F, 0x00 };
    DWORD dwSendLength, dwRecvLength;

    rv = SCardEstablishContext(SCARD_SCOPE_SYSTEM, NULL, NULL, &hContext);
    if (rv != SCARD_S_SUCCESS)
    {
        printf("SCardEstablishContext: Cannot Connect to Resource Manager%
            lX\n", rv);
        return EXIT_FAILURE;
    }

    /* Retrieve the available readers list. */
    dwReaders = SCARD_AUTOALLOCATE;
    rv = SCardListReaders(hContext, NULL, (LPSTR)&mszReaders, &dwReaders);
    PCSC_ERROR(rv, "SCardListReaders")

    /* Extract readers from the null separated string and get the total
     * number of readers */
    nbReaders = 0;
    ptr = mszReaders;
    while (*ptr != '\0')
    {
        ptr += strlen(ptr)+1;
        nbReaders++;
    }

    if (nbReaders == 0)
    {
        printf("No reader found\n");
        goto end;
    }

    /* Allocate the readers table */
    readers = calloc(nbReaders, sizeof(char *));
    if (NULL == readers)
    {
        printf("Not enough memory for readers[]\n");
        goto end;
    }

    /* Fill the readers table */
    nbReaders = 0;
    ptr = mszReaders;
    while (*ptr != '\0')
    {
        printf("%d: %s\n", nbReaders, ptr);
        readers[nbReaders] = ptr;
        ptr += strlen(ptr)+1;
        nbReaders++;
    }

    /* Read the reader number given in command line */
    if (argc > 1)
    {
        reader_nb = atoi(argv[1]);
        if (reader_nb < 0 || reader_nb >= nbReaders)
        {
            printf("Wrong reader index: %d\n", reader_nb);
            goto end;
        }
    }
    else
        reader_nb = 0;

    /* Connect to a card */
    dwActiveProtocol = -1;
    rv = SCardConnect(hContext, readers[reader_nb], SCARD_SHARE_SHARED,
        SCARD_PROTOCOL_T0 | SCARD_PROTOCOL_T1, &hCard, &dwActiveProtocol);
    printf("Protocol: %ld\n", dwActiveProtocol);
    PCSC_ERROR(rv, "SCardConnect")

    /* Get card status */
    dwAtrLen = sizeof(pbAtr);
    dwReaderLen = sizeof(pbReader);
    rv = SCardStatus(hCard, pbReader, &dwReaderLen, &dwState, &dwProt,
        pbAtr, &dwAtrLen);
    printf("Reader: %s (length %ld bytes)\n", pbReader, dwReaderLen);
    printf("State: 0x%lX\n", dwState);
    printf("Prot: %ld\n", dwProt);
    printf("ATR (length %ld bytes):", dwAtrLen);
    for (i=0; i<dwAtrLen; i++)
        printf("%02X", pbAtr[i]);
    printf("\n");
    PCSC_ERROR(rv, "SCardStatus")

    switch(dwActiveProtocol)
    {
        case SCARD_PROTOCOL_T0:
            pioSendPci = SCARD_PCI_T0;
            break;
        case SCARD_PROTOCOL_T1:
            pioSendPci = SCARD_PCI_T1;
            break;
        default:
            printf("Unknown protocol\n");
            goto end;
    }

    /* Exchange APDU */
    dwSendLength = sizeof(pbSendBuffer);
    dwRecvLength = sizeof(pbRecvBuffer);
    printf("Sending: ");
    for (i=0; i<dwSendLength; i++)
        printf("%02X ", pbSendBuffer[i]);
    printf("\n");
    rv = SCardTransmit(hCard, pioSendPci, pbSendBuffer, dwSendLength,
        &pioRecvPci, pbRecvBuffer, &dwRecvLength);
    printf("Received: ");
    for (i=0; i<dwRecvLength; i++)
        printf("%02X ", pbRecvBuffer[i]);
    printf("\n");
    PCSC_ERROR(rv, "SCardTransmit")

    /* Card disconnect */
    rv = SCardDisconnect(hCard, SCARD_LEAVE_CARD);
    PCSC_ERROR(rv, "SCardDisconnect")
end:
    /* Free allocated memory */
    if (mszReaders)
        SCardFreeMemory(hContext, mszReaders);

    /* We try to leave things as clean as possible */
    rv = SCardReleaseContext(hContext);
    if (rv != SCARD_S_SUCCESS)
        printf("SCardReleaseContext: %s (0x%lX)\n", pcsc_stringify_error(
            rv),
            rv);

    if (readers)
        free(readers);

    return EXIT_SUCCESS;
}
```

Listing 12.12 C language code sample

Perl Language

```
#!/usr/bin/perl -w

use Chipcard::PCSC;

#------------------------------------------------------------------------
# Create a new object print("Getting context!\n"); $hContext = new
Chipcard::PCSC(); if (!defined($hContext)){ die("Cannot create the
PCSC object:
                                                    $Chipcard::PCSC::errno\n")
                                                                ;}

#------------------------------------------------------------------------
# Retrieve the readers list
print("Retrieving the readers list:\n");
@ReadersList = $hContext->ListReaders();
if (!defined($ReadersList[0])) {die("Cannot get the readers list:
 $Chipcard::PCSC::errno\n");}
```

```
# Print the list of readers
$, = "\n  ";
$" = "\n  ";
print("   @ReadersList\n");


#------------------------------------------------------------------------
# Connect to the card ("the first reader")
print("Connecting to the card!\n");
$hCard = new Chipcard::PCSC::Card($hContext, $ReadersList[0]);
if (!defined($hCard)) {die("Cannot connect: $Chipcard::PCSC::errno\n");}

#------------------------------------------------------------------------
# Exchange APDU
print("Exchanging APDU:\n");
$capdu = Chipcard::PCSC::ascii_to_array("00 A4 00 00 02 3F 00");
print("--->".Chipcard::PCSC::array_to_ascii($capdu)."\n");
$rapdu = $hCard->Transmit($capdu);
if (!defined($rapdu)) {die("Cannot transmit or receive data:
                                    $Chipcard::PCSC::errno\n");}
print("<--- ".Chipcard::PCSC::array_to_ascii($rapdu)."\n");

#------------------------------------------------------------------------
# Card disconnect
print("Disconnecting the card!\n");
$hCard->Disconnect();
if (!defined($hCard)) {print("Cannot disconnect: $Chipcard::PCSC::errno\n");}
undef $hCard;

#------------------------------------------------------------------------
# Release the context
print("Releasing the context!\n");
$hContext = undef;
```

Listing 12.13 C language code sample

References

1. USB Implementers Forum. More Information Available via http://www.usb.org/about, Cited 14 Apr 2016.
2. OpenCard Framework. More Information Available via https://sourceforge.net/projects/opencard/, Cited 14 Apr 2016.
3. Uwe Hansmann, Martin S. Nicklous, Thomas Schck, and Frank Seliger. *Smart Card Application Development Using Java.* Springer, 2000.
4. Vesna Hassler, Martin Manninger, Mikhail Gordeev, and Christoph Muller. *Java Card for E-Payment Applications.* Artech House Publishers, 2001.
5. Java Smartcard I/O API. More Information Available via https://docs.oracle.com/javase/8/docs/jre/api/security/smartcardio/spec/javax/smartcardio/package-summary.html, Cited 14 Apr 2016.

6. Frank Seliger. *The OpenCard Framework and PC/SC – Two New Industry Initiatives for Smart Cards*, 1999. More Information Available via http://www.scardsoft.com/documents/OCF/OCF_PCSC.pdf, Cited 14 Apr 2016.
7. `javax.smartcardio` provider with less PC/SC restrictions. More Information Available via https://github.com/intarsys/smartcard-io, Cited 14 Apr 2016.
8. Project SCUBA. More Information Available via http://scuba.sourceforge.net/, Cited 14 Apr 2016.
9. A re-implementation of the `javax.smartcardio` API. More Information Available via https://github.com/jnasmartcardio/jnasmartcardio, Cited 14 Apr 2016.
10. A Java applet for smartcard-aware web pages. More Information Available via http://www.springcard.com/en/download/find/file/pmd0160, Cited 14 Apr 2016.
11. PC/SC Workgroup. More Information Available via http://www.pcscworkgroup.com/, Cited 14 Apr 2016.
12. PC/SC Workgroup. PC/SC Workgroup Specifications 1.0. More Information Available via http://www.pcscworkgroup.com/specifications/specdownloadV1.php, Cited 14 Apr 2016.
13. PC/SC Workgroup. PC/SC Workgroup Specifications 2.01.14. November 2013. More Information Available via http://www.pcscworkgroup.com/specifications/specdownload.php, Cited 14 Apr 2016.
14. pcsc-lite home page. More Information Available via http://alioth.debian.org/projects/pcsclite/, Cited 14 Apr 2016.
15. Microsoft. SCard API. More Information Available via https://msdn.microsoft.com/en-us/library/aa380149(v=vs.85).aspx, Cited 14 Apr 2016.
16. WHQL Testing - overview. More Information Available via http://www.microsoft.com/whdc/whql/, Cited 14 Apr 2016.
17. USB Implementers Forum. Universal Serial Bus Device Class Specification for USB Chip/Smart Card Interface Devices version 1.0, March 2001. More Information Available via http://www.usb.org/developers/docs/devclass_docs/DWG_Smart-Card_USB-ICC_ICCD_rev10.pdf, Cited 14 Apr 2016.
18. USB Implementers Forum. Universal Serial Bus Device Class Specification for USB Chip/Smart Card Interface Devices version 1.1, April 2005. More Information Available via http://www.usb.org/developers/docs/devclass_docs/DWG_Smart-Card_CCID_Rev110.pdf, Cited 14 Apr 2016.
19. Microsoft. CryptoAPI. More Information Available via https://msdn.microsoft.com/en-us/library/aa382404(v=vs.85).aspx, Cited 14 Apr 2016.
20. David Corcoran. *M.U.S.C.L.E: porting the PC/SC architecture to Linux.* Gemplus Developers Conference 99, June 1999.
21. MUSCLE home page. More Information Available via http://pcsclite.alioth.debian.org/musclecard.com/software.html, Cited 14 Apr 2016.
22. pcsc-lite SCard API. More Information Available via http://pcsclite.alioth.debian.org/pcsc-lite/, Cited 14 Apr 2016.
23. `pcsc-lite` Detailed Description. More Information Available via http://pcsclite.alioth.debian.org/api/group__API.html#details, Cited 14 Apr 2016.
24. Generic CCID IFD Handler home page. More Information Available via http://pcsclite.alioth.debian.org/ccid.html, Cited 14 Apr 2016.
25. PC/SC APDU inspection and manipulation tool. More Information Available via http://www.fi.muni.cz/~xsvenda/apduinspect.html, Cited 14 Apr 2016.
26. PCSC API spy, third try. More Information Available via http://ludovicrousseau.blogspot.fr/2011/11/pcsc-api-spy-third-try.html, Cited 14 Apr 2016.
27. PCSC API spy, on Mac OS X. More Information Available via http://ludovicrousseau.blogspot.fr/2014/02/pcsc-api-spy-on-mac-os-x.html, Cited 14 Apr 2016.
28. Perl Wrappers for PC/SC. More Information Available via http://ludovic.rousseau.free.fr/softwares/pcsc-perl/, Cited 14 Apr 2016.
29. Python Wrappers for PC/SC. More Information Available via http://pyscard.sourceforge.net/, Cited 14 Apr 2016.

30. GNU Prolog Wrappers for PC/SC. More Information Available via http://gprolog.cvs.sourceforge.net/gprolog/contribs/gplpcsc/, Cited 14 Apr 2016.
31. Ruby Wrappers for PC/SC. More Information Available via https://github.com/costan/smartcard, Cited 14 Apr 2016.
32. Java Wrappers for PC/SC. More Information Available via http://damien.sauveron.fr/software-developments/tools/, Cited 14 Apr 2016.
33. PC/SC sample in different languages. More Information Available via http://ludovicrousseau.blogspot.fr/2010/04/pcsc-sample-in-different-languages.html, Cited 14 Apr 2016.
34. Design Issues for Foreign Function Interfaces. A survey of existing native interfaces for several languages and some suggestions. More Information Available via http://autocad.xarch.at/lisp/ffis.html, Cited 14 Apr 2016.
35. OCF To PC/SC Shim. More Information Available via http://damien.sauveron.fr/software-developments/tools/, Cited 14 Apr 2016.
36. Welcome to the Virtual Smart Card Architecture documentation! More Information Available via http://frankmorgner.github.io/vsmartcard/, Cited 14 Apr 2016.
37. PC/SC Relay. More Information Available via https://frankmorgner.github.io/vsmartcard/pcsc-relay/README.html#pcsc-relay, Cited 14 Apr 2016.
38. Nikolay Elenkov. Android Security Internals. An In-Depth Guide to Androids Security Architecture. No Starch Press. 2015. ISBN: 978-1-59327-581-5.
39. Nikolay Elenkov. Accessing the embedded secure element in Android 4.x. More Information Available via http://nelenkov.blogspot.fr/2012/08/accessing-embedded-secure-element-in.html, Cited 14 Apr 2016.
40. Nikolay Elenkov. Using the SIM card as a secure element in Android. More Information Available via http://nelenkov.blogspot.fr/2013/09/using-sim-card-as-secure-element.html, Cited 14 Apr 2016.
41. SIMAlliance Open Mobile API Specification V3.2. More Information Available via http://simalliance.org/wp-content/uploads/2015/03/SIMalliance_OpenMobileAPI_v3_2.pdf, Cited 14 Apr 2016.
42. SEEK for Android. More Information Available via http://seek-for-android.github.io/, Cited 14 Apr 2016.
43. Michael Hölzl and Michael Roland. SEEK adaptations in RIL for Galaxy S3. More Information Available via https://usmile.at/blog/seek-galaxys3, Cited 14 Apr 2016.
44. SEEK – Emulator extension. More Information Available via https://github.com/seek-for-android/pool/wiki/EmulatorExtension, Cited 14 Apr 2016.
45. SIMAlliance Open Mobile API Specification V3.0. More Information Available via http://simalliance.org/wp-content/uploads/2015/03/SIMalliance_OpenMobileAPI3_0_release1_FINAL3.pdf, Cited 14 Apr 2016.
46. Oracle. Security and Trust Services API for J2ME (SATSA); JSR 177, 2016. More Information Available via http://www.oracle.com/technetwork/java/satsa-136426.html, Cited 14 Apr 2016.
47. Keith Mayes, Konstantinos Markantonakis. Secure Smart Embedded Devices, Platforms and Applications. Springer. 2014. ISBN 978-1-4614-7915-4.
48. bR301 Android API. More Information Available via https://github.com/FeitianSmartcardReader/bR301_Android/blob/master/Docs/bR301_Android_Developer_Guide_E.pdf, Cited 14 Apr 2016.
49. bR301 Android Samples. More Information Available via https://github.com/FeitianSmartcardReader/bR301_Android/tree/master/Sample%20code, Cited 14 Apr 2016.
50. bR301 & iR301 iOS API. More Information Available via https://github.com/FeitianSmartcardReader/bR301_iOS_SDK_RELEASE, Cited 14 Apr 2016.
51. bR301 & iR301 iOS Samples. More Information Available via https://github.com/FeitianSmartcardReader/bR301_iOS_SDK_RELEASE/tree/master/Sample, Cited 14 Apr 2016.
52. aR530 Android API. More Information Available via https://github.com/FeitianSmartcardReader/aR530-Audiojack-NFC-Reader/blob/master/aR530_SDK_V1.4_Latest/Android/Docs/aR530_Developer_Guide__Android_E.pdf, Cited 14 Apr 2016.

53. aR530 Android Samples. More Information Available via https://github.com/FeitianSmartcardReader/aR530-Audiojack-NFC-Reader/tree/master/aR530_SDK_V1.4_Latest/Android/Demo%20code/src/com/ft/mobile/reader/test, Cited 14 Apr 2016.
54. aR530 iOS Samples. More Information Available via https://github.com/FeitianSmartcardReader/aR530-Audiojack-NFC-Reader/tree/master/aR530_SDK_V1.4_Latest/iOS/Demo%20code/com.ftsafe.aR530.demo, Cited 14 Apr 2016.
55. aR530 iOS API. More Information Available via https://github.com/FeitianSmartcardReader/aR530-Audiojack-NFC-Reader/blob/master/aR530_SDK_V1.4_Latest/iOS/DOC/aR530_Developer_Guide_iOS_E_.pdf, Cited 14 Apr 2016.

Chapter 13
RFID and Contactless Technology

Anjia Yang and Gerhard P. Hancke

Abstract An increasing number of ‘contactless’ systems are based on passive Radio-Frequency Identification (RFID) technology. A passive RFID token is powered by a transmitted RF carrier, which is also used for bidirectional communication. RFID technology comprises of several standards, which are suitable for different applications. Electronic Product Code (EPC) tags, contactless credit cards, e-passports, and access control are just a few examples of systems that use a subset of this technology. This chapter contains a brief explanation of RFID operating principles along with an overview of prominent implementations and industry standards.

Keywords Radio Frequency Identification (RFID) · Contactless Cards · Near-Field Communication · Over the Air (OTA) · Deployment Modes · Secure Element · Relay Attack · Cloning · ISO 18092 · ISO 14443 · ISO 15693

13.1 Introduction

RFID is a technology that increases productivity and convenience and is currently being integrated into many areas of society. RFID is flexible and the variety of standards and tokens available allows it to be tailored to any application. The technology also offers additional benefits such as reduced maintenance cost and extended product lifetime. The uses of radio-frequency identification have therefore grown remarkably, and it is reported that the cumulative number of tokens sold from 1943 to the start of 2015 is about 34 billion [1]. The total RFID market, including systems and services, is valued at \$9.56 billion at 2015 and expected to increase to \$14.5 billion by 2020 [1].

Despite its recent popularity, RFID technology has been around for more than half a century. It is commonly believed that the concept of radio identification started in the early 1940s with the advent of radar when IFF transponders actively modulated the

A. Yang · G.P. Hancke (✉)
City University of Hong Kong, Hong Kong, China
e-mail: ghancke@ieee.org

A. Yang
e-mail: ayang3-c@my.cityu.edu.hk

K. Mayes and K. Markantonakis (eds.), *Smart Cards, Tokens, Security and Applications*, DOI 10.1007/978-3-319-50500-8_13

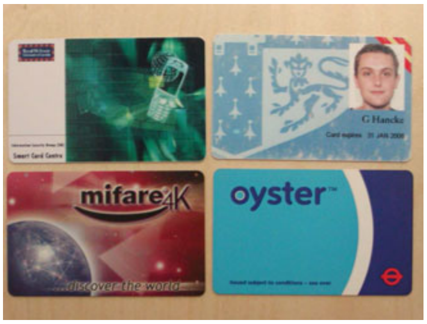
radiated ground radar signals to identify airplanes. Despite early work on RFID, such as Stockman's '*Communication by Means of Reflected Power*' in 1948 [2], recognising the potential of RFID, it would take several more years, and additional advances in electronics, before the technology was implemented in further applications. During the 1960s, stores and libraries used electronic article surveillance, an early 1-bit form of RFID, for theft control. Meanwhile, private and government research on the subject continued and in 1973 the first patents were filed that resembled modern systems: a token with rewritable memory by M.W. Cardullo and a passive token used to unlock doors by C. Walton. Scientists from the Los Alamos National Laboratory, who were asked by the US Energy Department to develop a tracking system for nuclear materials, also demonstrated the concept of modulated backscatter with 12-bit tokens operating at 915 MHz in the same year. The basic communication principle of this system is still used today by the majority of Ultra-High-Frequency (UHF) RFID tags. In the late 1980s, RFID gained widespread acceptance in automated toll collection and access control systems, which was followed by implementation in public transport payment systems and the first serious attempts at standardisation in the 1990s. In 1999, it was proposed that low-cost UHF RFID 'tags' could be used to track items in supply chains. Currently, the use of Electronic Product Code (EPC) tags for tracking at the pallet, case, and item level is probably the most prominent RFID application, driven by government agencies, such as the US Department of Defense, and various large retailers such as Tesco and Walmart. RFID technology is, however, also used on a large scale in other applications such as machine-readable travel documents (e.g., e-passports) and credit cards [3].

This chapter is intended as an introduction to RFID and summarises the aspects most relevant to contactless smart card systems. Section 13.2 gives a brief overview of existing systems, describing the general operating principles and available technology and discusses a number of high-profile implementations. The RF interface and communication theory are discussed in Sect. 13.3 and the current HF RFID standards are summarised in Sect. 13.4. The chapter concludes with an overview of Near-Field Communication (NFC) in Sect. 13.5.

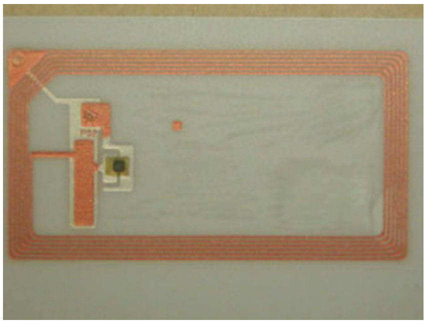
13.2 Contactless Technology

Even though RFID is a collective term for a number of technologies, it is often used primarily to describe applications using low-resource devices, such as EPC tags. Devices used in identity and payment systems, like the cards shown in Fig. 13.1, are instead regularly referred to as 'contactless' or 'proximity' tokens. These devices operate in the High-Frequency (HF) radio band, contain more resources, and have a much shorter operating range than their UHF counterparts.

Examples of contactless tokens are shown in Fig. 13.1a. Each token, or Proximity Integrated Circuit Card (PICC), contains an antenna and an Integrated Circuit (IC) as shown in Fig. 13.1b. The IC performs modulation and demodulation of the RF channel and is also responsible for data storage and processing. The passive token derives its power from the RF carrier transmitted by the reader, or 'Proximity Coupling



(a) Example tokens



(b) Example of an inlay, showing RFID IC and antenna

Fig. 13.1 ‘Contactless’ tokens

Device’ (PCD). The bidirectional communication between the token and reader is also modulated onto this carrier.

The main benefit of contactless technology is its ease of use. The user does not have to physically insert his token into the reader or orientate the token in a specific way. In most cases, the token can be kept in a wallet or a purse providing some measure of personal security. All these factors combine to provide fast transactions and ensure high throughput. Furthermore, readers and tokens have no external mechanical parts that can wear out. This makes systems more durable and reliable, especially in exposed or dirty environments, and reduces maintenance costs when compared to contact or magnetic stripe systems. The fact that the token does not need to contain a power supply adds to the lifetime of the token as well. Contactless tokens can also provide the same amount of security mechanisms than contact smart cards and there are several established international standards available to aid interoperability [4].

Contactless systems differ from each other in a number of ways. It is therefore important to note the alternatives offered by the available standards and products in order to decide on the best system components for a specific application. For example, the three HF standards (ISO 14443 [5], ISO 15693 [6], and ISO 18092 [7]) allow for different data rates and operational ranges. The growth in the contactless market has also resulted in a variety of readers and tokens. In most cases, the reader’s only purpose is to act as a RF transceiver between the back-end system, which performs all the processing and security functions, and the token. In general, the only difference between readers are the standards that they support. Readers are also more expensive and are often installed as a long-term infrastructure investment. It is therefore common to find readers supporting multiple standards so that the same hardware can be used in several applications, possibly allowing for future changes and extra functionality.

In contrast, there are quite a few tokens available that can be classified in terms of resources, security, and interfaces [4]. In terms of resources, tokens can be divided into three different types:

- **Memory**: These tokens only have the ability to store information. They can perform no processing in addition to read and write functions.
- **Logic**: In addition to memory, these tokens also include some fixed processing routines that can be invoked by the reader, e.g., authenticate, increment value, decrement value.
- μ-**Controller (MCU)**: The token can run custom processing routines and might contain card operating systems such as JCOP or MULTOS.

Tokens can implement a number of security mechanisms. Security increases the token's required resources resulting in a higher system cost, so it is important that the token used provides a level of security sufficient for the application without incurring unnecessary expense. These levels can roughly be defined as follows:

- **Minimal**: Anyone can read information stored on the token. Memory can be locked so no unauthorized writing of data occurs.
- **Low**: The token implements some form of authentication mechanism. Memory is password protected or mutual authentication must be completed before data is released.
- **Medium**: The token implements a single encryption algorithm used to provide authentication and encryption of data. The algorithm could be proprietary, e.g., NXP Crypto1, or an industry standard, e.g., Data Encryption Standard (DES).
- **High**: The token can implement a number of symmetric and asymmetric industry standard algorithms for authentication, encryption, digital signatures, etc.

It may be required that a token supports several technologies. This allows for an environment where the user can carry a single token to access multiple systems. Alternatively it provides a way to migrate to, or add, a new system while still maintaining backward compatibility with existing systems. The following tokens are available that have the ability to interact with more than one system:

- **Multiple-Technology**: The token implements multiple technologies. A good example, albeit not contactless, of this is Chip and Pin cards in the UK that contain both magnetic stripe and contact smart card technologies.
- **Dual-Interface**: The token contains one integrated circuit that has more than one interface. An example would be a token containing an Integrated Circuit (IC) with both a contactless and a contact interface.
- **Hybrid**: A token with two or more integrated circuits, with their own interfaces, functioning independently. This term can be used to describe HF contactless tokens that also contain additional circuitry to support older Low-Frequency (LF) systems.

13.2.1 Applications

Contactless tokens act as an electronic credential, interacting with the rest of the system on behalf of the entity it is associated with. Initially, these tokens allowed for new applications such as contactless access control and automatic toll collection. In

Table 13.1 Summary of HF RFID tokens applications

Application	Standard	Token resources	Security
Item tracking	ISO 15693	Memory	Minimal/low
Ticketing	ISO 15693 ISO 14443	Memory/logic	Low/medium
Closed payment	ISO 14443	Logic	Low/medium
Open payment	ISO 14443	μ-controller	High
Access control	ISO 14443	μ-controller	High
Identity	ISO 14443	μ-controller	High

recent years, however, these tokens have started to replace, or supplement, established technologies such as paper tickets for travel or events, barcodes in item tracking and magnetic stripes in credit cards. This section gives an overview of prominent applications in which contactless tokens are used. Table 13.1 summarises some of these applications.

13.2.1.1 Identification

The basic function of RFID, as the name already suggests, is to assist a system in uniquely identifying an item or a person. The simplest example of this is *tracking* systems where a token, storing a Unique Identifier (UID), is attached to an item. The system can then track this item by scanning the token every time it passes a reader. Systems using HF tokens have a shorter operational range than their UHF equivalents although they provide more reliable reader coverage. NXP I-CODE and the Texas Instruments Tag-It products are examples of HF tokens used in tracking applications. In *ticketing* systems, tokens facilitate access to services for a limited time before being disposed of. A ‘ticket’ can take many forms, such as a key card to a hotel room or a day pass at the local gym. Paper tickets, containing NXP Mifare UltraLight tokens, received extensive publicity during the FIFA World Cup in 2014 and were used to gain fast access to stadiums and enhance personal security by tracking customers going off to remote locations such as ski slopes [8].

Contactless *access control* is popular for securing physical locations. Charles Walton first invented an RFID-based access control system in 1973. The system involved an electronic lock that opened with an RFID key card, which he sold to Schlage [3]. Since then, contactless access control systems have become widespread not only in the private sector but also with government agencies. An application closely linked to access control is that of *identity*. A token used for proof of identity must contain enough information to allow the system to verify that the person presenting it is the legitimate owner. Identity tokens therefore contain additional personal information, such as biometric data.

As an example, the Federal Information Processing Standard Publication 201 (FIPS 201) detailing Personal Identity Verification (PIV) of federal employees and contractors [9] is published by the US National Institute for Standards and Technology (NIST) at 2006 and has been updated to FIPS 201-2 [10] at 2013. It provides specifications for a standard Federal smart ID card that is to be used for access control. The cards are a requirement for all federal employees and contractors under the Homeland Security Presidential Directive 12 (HSPD-12). Other large government initiatives in the USA include:

- The Department of Defense's Common Access Card with Contactless (CAC-C) being used as identification for on duty military personnel, reserve personnel, and civilian employees.
- The Transportation Worker Identification Credential (TWIC) being issued by the Transportation Security Administration.
- The First Responder Authentication Card (FRAC) being issued in the Department of Homeland Security (DHS) pilots [11].

Contactless tokens are also used for national identity cards and some countries are planning to use contactless ID cards for their citizens, with China becoming the largest implementer. The most prominent application of contactless identity is, however, Machine-Readable Travel Documents (MRTD). By the time of writing, the USA required that 38 countries issue their citizens with e-passports in order to still qualify under the Visa Waiver Program. E-passports adhere to operational specifications as defined by the International Civil Aviation Organization (ICAO) [12], with additional guidelines for MRTD specified in ISO 7501 [13]. By 2015, ICAO wishes to have replaced all current passports with a digital version that stores encrypted biometric data on a RFID chip. The DHS also wants to use passive RFID to record who is entering or leaving the USA across land routes using a People Access Security Service (PASS) card. ICAO allows for optional security protocols that provides both authentication and encryption services. E-passports have the interesting security requirement that anyone who is presented with the passport should be able to read and verify the contents, but at the same time the user's personal details should be afforded some measure of privacy. For this reason, most e-passports implement the Basic Access Control (BAC) scheme. BAC derives a key from the passport number, expiry date, and the user's birthday, read off the Optical Character Recognition (OCR) strip inside the passport. The idea is that anyone legitimately presented with the passport can read the OCR data, derive the key and retrieve the data off the token inside. The European Union countries have already implemented Extended Access Control (EAC), involving a public key infrastructure for participating parties, for future passports including additional biometric data [14].

13.2.1.2 Payment

RFID has been used in payment systems since the 1980s, when it was first used for automatic toll collection. Since then, contactless tokens were implemented in several cashless payment systems. In *closed* systems, one organisation is in control of the entire payment process. In other words, customers will pay for the organisation's services with payment tokens issued by the same organisation. These systems often function on a 'prepaid' principle where the customer pays for credit in advance and are ideal for public transport payments. A good example of such a system is the Oyster card scheme implemented by Transport for London (TFL) using NXP Mifare Classic tokens. Closed payment systems can also be used in other environments such as service stations, e.g., the SpeedPass system implemented by ExxonMobil, or in some cafeterias and fast food outlets.

In *open* systems, an organisation issues customers with payment tokens that will be used to purchase services from other organisations. Some of these systems still operate using prepaid credit like Hong Kong's Octopus system, implemented with Sony Felica tokens. The Octopus card is not only used mainly for transport payments but can also be used to pay at convenience stores, restaurants, and other local services. The most prominent open payment system is, however, contactless credit cards. RFID credit cards have been widely deployed in the USA [15] with American Express (ExpressPay), MasterCard (PayPass), and Visa (payWave) all supporting contactless credit and debit cards.

In the UK, the Royal Bank of Scotland, working with Mastercard, and Barclays launched contactless debit cards in 2007. In particular, the Barclays' OnePulse card, developed with Visa, is intended to function as a chip-and-pin contact card, contactless debit card, and an Oyster card. The lower level communication of these cards, as specified EMV contactless specification [16] adheres to ISO 14443 while the application layer communication adheres to the same Europay, MasterCard, and Visa (EMV) framework and specifications as contact payment cards and transaction terminals [17].

13.3 Radio-Frequency Interface

The operation of contactless systems is based on the principle of inductive coupling. The token and the reader both contain antenna coils that are coupled and interact via a magnetic field. The token receives both data and power from the carrier transmitted by the reader. The token can also send data to the reader by influencing this carrier. Collectively, these methods are referred to as near-field communication since the range between the token is much smaller than one wavelength of the carrier. This section provides an overview of communication theory and physics relevant to contactless systems. The information in this section has been adapted from [18–20].

13.3.1 Communication Theory

In order to transmit information over an RF channel, the relevant data must first be encoded and then modulated onto a suitable RF carrier. Line, or data, coding changes the binary data into a signal sequence that is best suited to the transmission channel and aids the receiver in recovering the data. Modulation is the process whereby the parameters of an RF carrier is altered in relation to the resultant baseband signal. Modulation, and to a lesser extent coding, techniques can be used to shape the frequency spectrum of the communication channel. In contactless systems, coding and modulation methods have two main prerequisites: to separate the 'weak' backward channel communication from the strong forward channel carrier and to allow for data transfer from the reader to the token while ensuring that the token still receives adequate power from the HF carrier.

A line code's properties will typically be chosen to allow for the physical requirements of the transmission channel. Line codes are most often used to eliminate the DC component of the data and help the receiver with synchronisation, although some codes also provide redundancy to prevent errors. HF RFID tokens use the schemes shown in Fig. 13.2.

Non-Return-to-Zero (NRZ) coding is the most used of basic coding techniques. A '1' is represented by a logical high for one clock period and a '0' is represented by a logical low for one clock period. NRZ encoding is not ideal when used to transmit data without a maximum runlength constraint, or in other words data that contains long sequences of ones or zeros. In this case, there will be no signal transitions between high and low for a period of time, which can prevent the recovery of an accurate data clock. This also has other consequences, e.g., if a long sequence of zeros is transmitted using 100% amplitude modulation, it could disrupt the token's power supply.

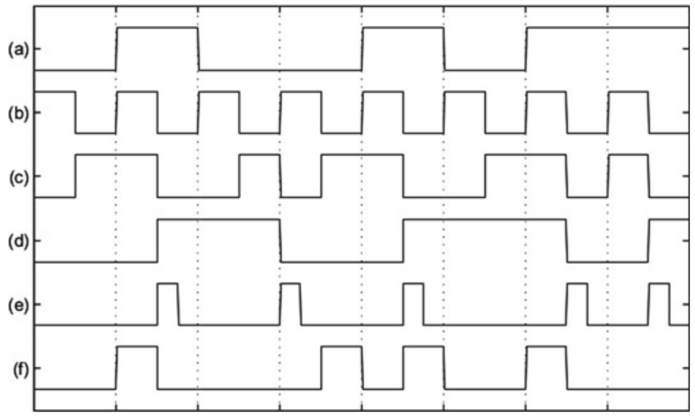


Fig. 13.2 Data coding examples: **a** Non-Return-to-Zero (NRZ), **b** data clock, **c** Manchester, **d** Miller, **e** Modified Miller, and **f** pulse-position

Manchester coded sequences have at least one transition during each bit period. This periodic transition, which occurs in the middle of the bit period, can therefore be used to recover the data clock regardless of the data values. A '0' is expressed by a low-to-high transition and a '1' by a high-to-low transition. Although Manchester encoded sequences contain no DC component, it requires approximately twice the channel bandwidth of NRZ.

Modified Miller coding is often used for data transmission from the reader to token. The data is encoded using Miller coding after which each transition, high-to-low and low-to-high, is replaced by a single pulse. In Miller encoding, a '1' is represented by a bit period with a transition at the midpoint. A '0', if preceded by a '1', is represented by a bit period with no transition while a '0', preceded by another '0', is represented by a bit period with a transition at the start of the bit period. The advantage of Modified Miller coding is that the carrier is not interrupted for more than the duration of the coding pulse, even if a bit period is very long. This ensures a continuous power supply to the token from the reader's carrier during data transfer. The bandwidth of Modified Miller coding depends on the duration of the coding pulse.

Pulse-Position coding, which is also sometimes referred to as Pulse-Position Modulation (PPM), represents x data bits with a single pulse in one of 2^x possible time slots. In Fig. 13.2, an example of '1 in 4' PPM is given. In this case, two bit periods are divided into four time slots. The data bits are then encoded as follows: '00' is represented with a pulse in the last slot, '01' with a pulse in slot three, '10' with a pulse in slot 2, and '11' with a pulse in the first slot. PPM is also suitable for reader to token communication since the carrier is not interrupted for more than the duration of the coding pulse. As with Modified Miller coding, the required bandwidth is determined by the time width of the coding pulse.

Modulation is the process whereby a RF carrier's parameters are changed to represent a baseband data sequence. A sinusoidal carrier can be characterised by

$$x(t) = a \cdot \sin(2\pi f_c t + \phi) \tag{13.1}$$

From the equation, it is clear that the amplitude a, frequency f_c, and phase ϕ can be varied to create distinctive carriers that are suitable to represent different data symbols. The amount that the chosen variable changes in relation to the data is referred to as the modulation index m_i. For example, m_i for amplitude modulation can be represented as

$$\frac{a_{\max} - a_{\min}}{a_{\max} + a_{\min}} \tag{13.2}$$

When modulating digital data, the chosen variable will only change between a set number of discrete values, e.g., high signal represented by $f = 10\,\text{Hz}$ and low signal represented by $f = 20\,\text{Hz}$. As a result, modulation is often referred to as 'shift keying' in digital systems. Examples of the modulation schemes used in HF RFID are shown in Fig. 13.3.

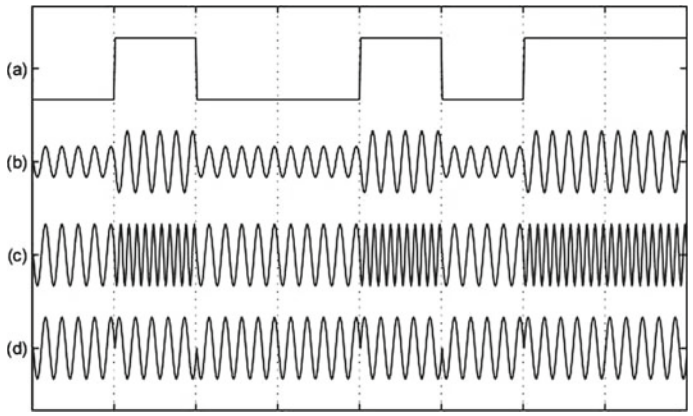


Fig. 13.3 Examples of RF modulation: **a** NRZ encoded data, **b** Amplitude-Shift Keying (ASK), **c** Frequency-Shift Keying (FSK), and **d** Phase-Shift Keying (PSK)

Amplitude-Shift Keying (ASK) changes the amplitude of the carrier to a level chosen to represent a specific data symbol. In the example shown, a '1' is represented by a carrier with amplitude A and a '0' is represented by a carrier with amplitude $0.5A$. In this case, the modulation index would be $0.33 \approx 33\%$. On–Off Keying (OOK) is a special case of ASK where the modulation index is equal to 100%.

Frequency-Shift Keying (FSK) changes the frequency of the carrier to represent a specific data symbol. In the example shown, a '0' is represented by a carrier with $f_c = f_1$ and a '1' is represented by a carrier with $f_c = 2 \cdot f_1$.

Phase-Shift Keying (PSK) represents each specific data symbol by a shift in the carrier's phase. In the example shown, a '0' is represented by a carrier with phase equal to 0° and a '1' is represented by a carrier with phase equal to 180°.

The modulation process also changes the frequency-domain representation of the data. ASK and PSK cause the power spectrum of the data to move from the baseband to f_c, while the spectrum power components for FSK data, as per our example, will be at f_1 and $2 \cdot f_1$.

The modulation process can be represented mathematically as follows:

$$x(t) = d(t) \cdot \sin(2\pi \cdot f_c t)$$
$$X(f) = D(f) * (\delta(-f_c) + \delta(f_c))$$
$$X(f) = D(f + f_c) + D(f - f_c)$$

where $d(t)$ is the baseband data sequence with frequency spectrum $D(f)$.

This is useful in RFID systems where both the forward and backward channel data are transmitted using the same carrier. The signal power of the backward channel can be up to 80 dB smaller than that of the carrier, which means that the reader would find it difficult to distinguish this 'weak' data from the 'strong' carrier if it cannot effectively isolate the data of interest and attenuate the carrier. Separating the two channels in the frequency domain simplifies the recovery of the backward channel

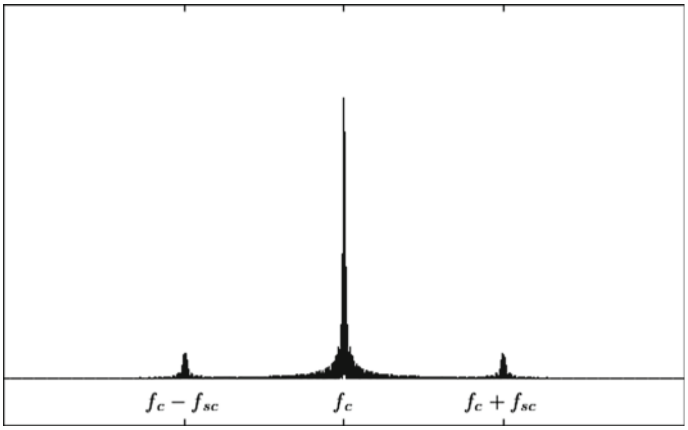


Fig. 13.4 The theoretical positive-frequency spectrum of the forward and backward channel modulated using a carrier with frequency f_c and a sub-carrier with frequency f_{sc}

data. In HF RFID, the forward channel data is modulated directly onto the main carrier. The backward channel, however, is first modulated onto a sub-carrier before being modulated onto the main carrier. Modulating with a sub-carrier creates two data sidebands separated by f_{sc} from the operational frequency f_c. The backward channel's modulation process can be represented mathematically as follows:

$$x(t) = (d_B(t) \cdot \sin(2\pi \cdot f_{sc}t)) \cdot \sin(2\pi \cdot f_c t)$$
$$X(f) = (D_B(f) * (\delta(-f_{sc}) + \delta(f_{sc}))) * (\delta(-f_c) + \delta(f_c))$$
$$X(f) = D_B(f + f_c + f_{sc}) + D_B(f + f_c - f_{sc}) + D_B(f - f_c + f_{sc}) + D_B(f - f_c + f_{sc})$$

where $d_B(t)$ is the backward channel data with frequency spectrum $D_B(f)$. The resultant positive-frequency spectrum of the forward and backward channels is shown in Fig. 13.4. The data at $f_c + f_{sc}$ is referred to as the upper-sideband while the data at $f_c - f_{sc}$ is referred to as the lower-sideband. The backward channel data can now be recovered, despite the presence of a strong operational carrier, by bandpass filtering one of the sidebands.

13.3.2 Inductive Coupling

HF RFID systems work on the principle of inductive coupling, where one device transfers energy to another by means of a shared magnetic field (H). This operational model is true as long as the token is placed in the near field of the reader, since the high-frequency electromagnetic field (E) generated by the reader acts primarily as a magnetic field if the distance between the reader and token is less than $\lambda_{f_c} \cdot \frac{1}{2\pi}$. In simple terms, an RFID system acts in a similar same way to a transformer, with the primary coil contained in the reader and the secondary coil contained in the token. Current flowing through the reader's coil generates a magnetic field, which in turn induces a proportional current flow in the coil of the token. The high-frequency carrier can therefore be used for both data and power transfer between the reader and the token.

13.3.2.1 Power Transfer

Moving charge, such as the flow of current in a conductor, generates a magnetic field. The magnitude of this field at a specific point is described by the magnetic field strength H. In HF RFID systems, the reader uses conductor loops to generate a magnetic field. The magnitude of the magnetic field generated by these loop antennas depends on the current I, number of loops N, the radius of the loops R, and the distance from the loop antenna d. The following equation can be used to calculate the field strength along the axis of the loop antenna:

$$H = \frac{I \cdot N \cdot R^2}{2\sqrt{(R^2 + d^2)^3}} \tag{13.3}$$

In this case, d is the distance from the centre of the coil along the coil's axis. In general, the field strength is almost uniform at short distances (d $<$ R) where smaller loop antennas also have a higher field strength. The larger antennas, however, have higher field strength at greater distances. A token will specify the minimum field strength it needs to function. Designing the antenna is then a trade-off between making the antenna small enough to generate a strong enough magnetic field and also making it large enough to achieve the system's required operational range.

In an HF RFID system, both the reader and the token contain loop antennas in close proximity, as shown in Fig. 13.5. If a second loop antenna, with area A_2, is located close to another loop antenna, with area A_1, then the second antenna will be affected by a proportion of the total magnetic flux Ψ flowing through the first antenna. The two circuits are connected together by this partial transfer of flux and is therefore said to be coupled. The magnitude of the coupled flux Ψ_{21} depends on the characteristics of the loop antennas, the position of the antennas in relation to each other and the magnetic conductivity, or permeability, of the medium between the antennas.

The ratio of the total flux that is generated in an area, to the current in the conductor that encloses that area, is described by inductance L:

$$L = \frac{\Psi}{I} = \frac{N \cdot \mu \cdot H \cdot A}{I} \tag{13.4}$$

The constant μ is equal to $\mu_0 \cdot \mu_r$, where μ_0 is the magnetic field constant ($4\pi \times 10^{-6}$) and describes the permeability of a vacuum, while μ_r is the relative permeability, indicating the ratio of the permeability of a material relative to μ_0.

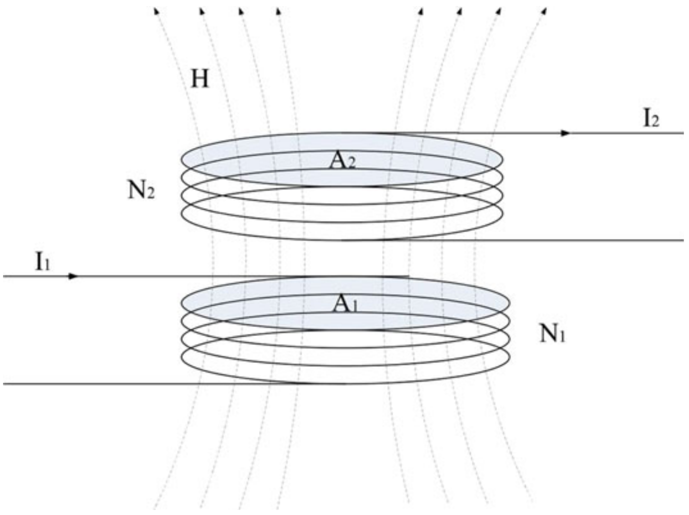


Fig. 13.5 Inductive coupling

The concept of mutual inductance is used to describe the coupling of two antennas by means of a magnetic field. The mutual inductance M_{21} is defined as the ratio of coupled flux Ψ_{21} enclosed by a second loop antenna to the current I_1 in the first loop:

$$M_{21} = \frac{\Psi_{21}(I_1)}{I_1} \tag{13.5}$$

Similarly, there is also a mutual inductance M_{12} although $M_{21} = M_{12} = M$. The mutual inductance between two loop antennas can be calculated using Eqs. 13.4 and 13.5 and can be approximated as follows:

$$M_{12} = \frac{\mu_0 \cdot H(I_1) \cdot N_2 \cdot A_2}{I_1} \tag{13.6}$$

Replacing $H(I_1)$ with Eq. 13.3 and substituting $R^2\pi$ for A the final result is:

$$M_{12} = \frac{\mu_0 \cdot N_1 \cdot R_1^2 \cdot N_2 \cdot R_2^2 \cdot \pi}{2\sqrt{(R_1^2 + d^2)^3}} \tag{13.7}$$

In practice, the HF carrier transmitted by the reader is a sinusoidal alternating current. A time-variant current $I_1(t)$ flowing in a loop antenna generates a time-variant magnetic flux $\mathrm{d}\Psi(I_1)/\mathrm{d}t$. According to Faraday's law, a voltage will be induced in the loop antenna that encloses some of this changing flux. Figure 13.6 shows a simplified circuit diagram for a coupled RFID system, where L_1 is the

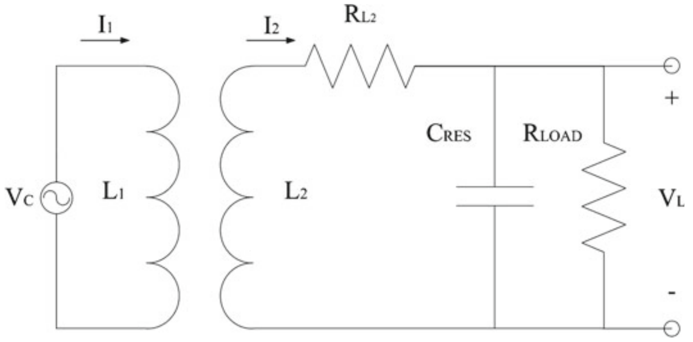


Fig. 13.6 Simplified circuit diagram of coupled token

antenna of the reader, L_2 is the antenna of the token, R_{L_2} is the resistance of the token's antenna, and R_{LOAD} represents the load. C_{RES} is ignored for now and is equal to 0.

The total time-variant flux in L_1 induces a voltage V_{L1} in L_2 due to the mutual inductance M. There is a voltage drop across R_{L_2} and I_2 also induces magnetic flux in L_2, which opposes $\Psi(I_1)$, so the voltage across the load can be approximated by:

$$V_L = \frac{\mathrm{d}\Psi_2}{\mathrm{d}t} = M\frac{\mathrm{d}I_1}{\mathrm{d}t} - L_2\frac{\mathrm{d}I_2}{\mathrm{d}t} - I_2R_{L_2} \tag{13.8}$$

V_L can now be rectified and used as a power supply for the token. In order to improve the efficiency of the coupling, an additional capacitor C_{RES} can be added in parallel with the antenna L_2 to form a parallel resonant circuit with a resonant frequency corresponding to the operating frequency of the RFID system. The resonant frequency can be calculated as follows:

$$f = \frac{1}{2\pi\sqrt{L_2 \cdot C_{RES}}} \tag{13.9}$$

When operating at the resonant frequency, the voltage V_L induced in a system with a resonant circuit increases by more than a factor of ten compared to a system using the antenna by itself. The influence of the resonant circuit's R_{L_2} and R_{LOAD} on voltage V_L can be characterised by the Q, or quality, factor. The Q-factor is discussed in more detail later with regard to data transfer.

A further practical constraint that effects the coupling efficiency is the orientation of the token in relation to the reader. In the equations above, it was assumed that the antenna in the reader and the antenna in the token have a common axis. Although this is often the case, the token can also be displaced or tilted in such a way that the magnetic flux enclosed by its antenna is decreased. As a rule of thumb, maximum voltage is induced in the token's antenna when it is perpendicular to the magnetic field lines, while no voltage is induced if it is parallel. This results in a specific interrogation

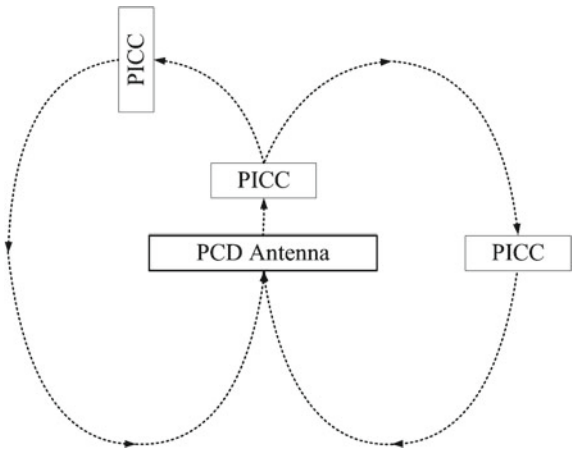


Fig. 13.7 Orientation of token-to-reader antenna for maximum coupling

zone around the reader's antenna, as illustrated by an example in Fig. 13.7. Following the expected path of the field lines in this case, it can be seen that a token parallel to the reader's antenna would be read if directly in front, or to the side, of the antenna, while a token that is perpendicular to the antenna could be read on the diagonal corners.

13.3.2.2 Data Transfer

The reader and token use different techniques to transmit data. As a result reader-to-token communication is referred to as the forward channel, while token-to-reader communication is referred to as the backward channel. The forward channel is relatively simple as the reader can directly modulate the data onto the carrier it transmits. The backward channel, however, requires that the token send data even though it is a passive device. The token must therefore modulate the reader's carrier, which in most cases is done using *load modulation*.

Load modulation works on the principle that the token's impedance Z_T can be altered by changing the parameters of the resonant circuit. Changing the impedance not only influences the voltage induced in the token's antenna L_2 but also the magnitude of the voltage across the reader's antenna L_1. The token can therefore amplitude modulate the voltage on the reader's antenna. It is only possible for the token to alter the load resistance R_{LOAD} or the parallel capacitor C_{RES} of the resonant circuit. Load modulation is therefore either resistive or capacitive. In resistive load modulation, a resistor R_{MOD} is added in parallel to R_{LOAD}, so the impedance is switched between $Z_T(R_{LOAD})$ and $Z_T(R_{LOAD}||R_{MOD})$ during the modulation process. In capacitive load modulation, an additional capacitor C_{MOD} is added, which changes the resonant frequency when switched into the circuit. Detuning the resonant circuit greatly influences the token's impedance, which causes the desired modulation effect.

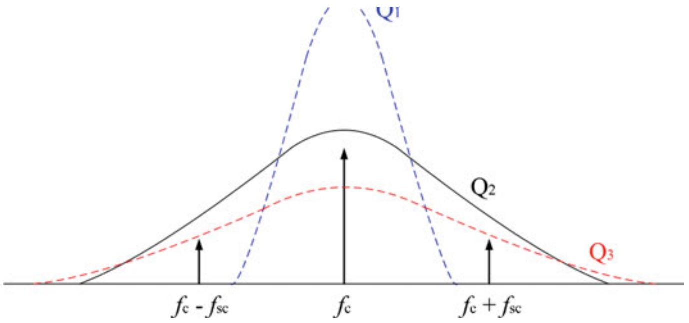


Fig. 13.8 The effect of the Q-factor

As mentioned before, the Q-factor is a measure of the voltage in the token when operating at the resonant frequency. It can be approximated as follows:

$$Q = \left(\frac{R_{L_2}}{2\pi f_c L_2} + \frac{2\pi f_c L_2}{R_{LOAD}} \right)^{-1} \tag{13.10}$$

Generally, the value of Q should be maximised to allow for the maximum power transfer and operational range. It should be kept in mind that Q also influences the bandwidth of the communication channel. The frequency response of Q peaks around the resonant frequency and rolls off to either side, as shown by the examples in Fig. 13.8. This indicates that the resonant circuit acts as a crude bandpass filter centered around f_c with bandwidth $BW = f_c/Q$. The value of Q must therefore be chosen in such a way that it allows sufficient power transfer, while still allowing for the backward channel's modulation sidebands. In Fig. 13.8, the value of Q_1 is too high since it excludes the modulation side bands, whereas the value of Q_3 is too low since it decreases the center frequency needed for power transfer.

13.4 Standards

RFID technology encompasses a range of systems from multiple vendors. To ensure interoperability between different RFID systems, several standards have been defined to which these systems must adhere. In the HF band, there are three main standards acknowledged by ISO/IEC (International Organization for Standardization/International Electrotechnical Commission) that deal with HF RFID technology operating at 13.56 MHz. These are ISO/IEC 14443 [5], ISO/IEC 15693 [6] and finally ISO/IEC 18092 [7]. Another standard that should be mentioned is ISO/IEC 18000, which defines RFID communication interfaces for several operating frequencies, including 13.56 MHz. The standards define, among other things, the RF interface, the initialization sequence, and the data format. It is not feasible to describe each standard in its entirety within this chapter, so only a summary of the key technical aspects of the interaction between the reader and the token is presented. ISO 14443 and ISO 15693 are discussed in more detail since it is the most relevant to the applications described in Sect. 13.2.1, with ISO 18000 and ISO 18092 discussed only briefly. The information in this section has been adapted from [5–7, 18, 21] and the reader should consult these sources for a more complete description.

13.4.1 ISO 14443

ISO 14443, titled *Identification Cards—Proximity Integrated Circuit Cards*, is commonly used in systems using logic and μ-controller tokens. This means that it is the standard of choice for e-passports, credit cards, and most access control systems. The popular Mifare range of products by NXP also adhere to Part 1–3 of ISO 14443 Type A.

13.4.1.1 Part 1—Physical Characteristics

The first part of the standard states that the token shall have physical characteristics according to the requirements for the card type ID-1 specified in ISO/IEC 7810, i.e., 85.72 mm × 54.03 mm × 0.76 mm. It also specifies tolerance levels for the token with regard to ultraviolet light, X-rays, dynamic bending stress, dynamic torsional stress, alternating magnetic fields, alternating electric fields, static electricity, static magnetic fields, and operating temperature.

13.4.1.2 Part 2—Radio-Frequency Power and Signal Interface

The second part of the standard describes the signal characteristics of two types of radio interfaces between the token and the reader. These interfaces allow for both power transfer and bidirectional communication. The token's power is provided by an alternating magnetic field at a frequency of 13.56 MHz and the reader must ensure that the magnetic field is within the range $1.5\,\text{A/m} \leq H \leq 7.5\,\text{A/m}$. The operational range for systems using ISO 14443 usually extends up to 10 cm.

ISO 14443 defines two different methods for data transfer. In *Type A*, the forward channel data uses Modified Miller coding with a coding pulse of 2–3 μs modulated onto the 13.56 MHz carrier with 100% ASK. The backward channel uses Manchester encoding, which is 100% ASK modulated onto a 847 kHZ sub-carrier before being load modulated onto the 13.56 MHz carrier. In *Type B*, the forward channel uses NRZ encoding modulated onto the 13.56 MHz carrier with 10% ASK. The backward channel uses NRZ encoding, which is first modulated onto a 847 kHZ sub-carrier using PSK (0°, 180°) before being load modulated onto the 13.56 MHz carrier. The basic data rate for both the forward and backward channels in Type A and Type B is

106 kbps. Some ISO 14443 tokens and readers support higher data rates, 212/424/848 kbps, which can be selected after the anti-collision process has finished.

13.4.1.3 Part 3—Initialization and Anti-collision

This part of the standard describes byte and data frame formats and the initial commands used to detect and initialise communication with the token. This includes the ability to poll for new tokens in the interrogation field and choosing a token, even if multiple tokens are present, through an anti-collision process.

Type A: A data frame is identified by Start-of-Frame (SoF) and End-of-Frame (EoF) symbols and may contain multiple bytes. Each byte of data is followed by an odd-parity bit. Each frame also contains a 2-byte Cyclic Redundancy Check (CRC) that is appended at the end of the data. During the initialization phase, the token acts like a state machine. The reader then issues commands to change the state of the token as required. A simplified Type A state machine is shown in Fig. 13.9. The reader uses the following commands:

- *REQA/WUPA*: The Type A Request (*REQA*) and the Type A Wake-Up (*WUPA*) commands are periodically sent by the reader to poll its interrogation field for tokens. The *WUPA* command can also be used to put tokens that have entered the HALT state into the READY state.

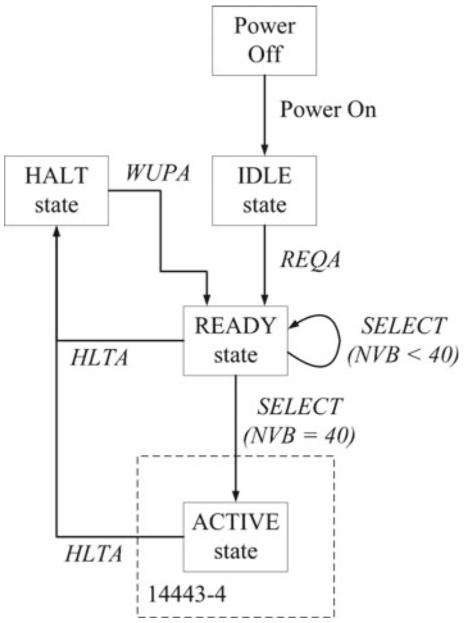


Fig. 13.9 Type A: token state machine

- *SELECT (NVB < 40)*: If the Number of Valid Bits (NVB) is less than the number of bits in the Unique Identifier (UID), then the *SELECT* command is used for anti-collision; thus, it is also referred to as the *ANTICOLLISION* command.
- *SELECT (NVB = 40)*: When the reader has determined the unique identifier of the token, it wishes to communicate with it using the *SELECT* command to place that token in the ACTIVE state.
- *HLTA*: The Type A Halt (*HLTA*) puts the token into the HALT state.

When the token enters the interrogation zone of the reader, it powers up and enters the IDLE state. In this state, the token will not respond to any commands from the reader except *REQA* and *WUPA*, which will put it into the READY state. Readers will periodically transmit a *REQA* command to see whether there are tokens within its interrogation zone. If it receives a Type A Answer to Request (*ATQA*) response, it knows that a token is present and will proceed to the next step of initialisation. It is often the case that multiple tokens will be presented to the reader at once, e.g., travel and credit cards in the same wallet, so the standard must allow the reader to select a specific token. This process of selection is known as anti-collision, which in Type A is implemented using a binary search tree algorithm. The *SELECT* command is a bit-oriented frame containing the length of the current search (NVB) and a search pattern. If the least significant bits of a token's unique identifier match the search pattern for the specified search length, that token will respond with the rest of its unique identifier. An example of the anti-collision process is shown in Fig. 13.10. In the first step, the reader sets the search length to 0, which results in both tokens transmitting their whole 4-byte unique identifier and a checksum (BCC). The first collision that the reader detects is in the fifth bit. The reader therefore sets the search length to 5 and transmits a search pattern that indicates to the token whether the fifth bit should be a '1' or a '0'. Only the first token's unique identifier matches the search

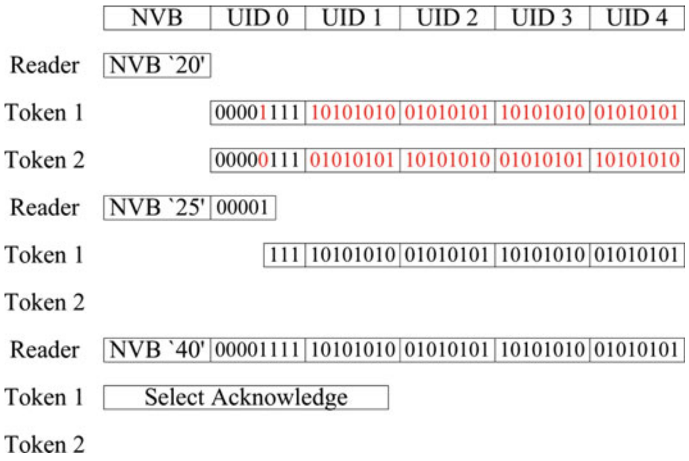


Fig. 13.10 Type A: example of anti-collision sequence

string, so it alone responds with the rest of its identifier. Since the reader detects no collisions, it knows that it has identified a single token. Finally, the reader sends the same *SELECT* command with maximum search length and full unique identifier to which the tokens respond with a Select AcKnowledge (SAK). This also results in the token being put in the ACTIVE state.

For the anti-collision to work, the token's responses must be closely synchronised with the reader's commands. Type A expects the token to behave in a synchronous manner, so it prescribes a fixed bit grid which defines when a response must be sent. This grid is defined by specifying a Frame Delay Time (FDT) as follows: FDT $= (n \cdot 128 + 84)/f_c$ if the last bit sent by the reader is a '1' and FDT $= (n \cdot 128 + 20)/f_c$ if the last bit sent by the reader is '0'. n is equal to 9 for *REQA*, *WUPA* and *SELECT* commands, while $n \geq 9$ for all other commands.

Type B: A data frame, marked with start-of-frame and end-of-frame symbols, contains multiple characters. Each character consists of a start bit, eight data bits, and a stop bit, which must be followed by a set Extra Guard Time (EGT) before the next character starts. Each frame also contains a 2-byte cyclic redundancy check that is appended at the end of the data.

As with Type A, the token acts like a state machine during initialisation. A simplified Type B state machine is shown in Fig. 13.11. The reader uses the following commands:

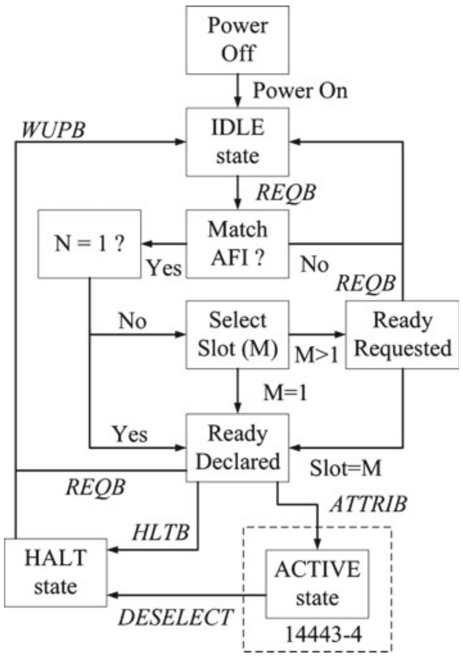


Fig. 13.11 Type B: token state machine

- *REQB/WUPB*: The Type B Request (*REQB*) and the Type B Wake-Up (*WUPB*) commands are periodically sent by the reader to poll its interrogation field for tokens and initiates the anti-collision procedure. The *WUPB* command can also be used to put tokens that have entered the HALT state into the IDLE state. *REQB* and *WUPB* commands contain an Application Family Identifier (AFI) that indicates the type of application targeted by the reader. This field is used to preselect tokens participating in the anti-collision since only tokens with an application of the type indicated by the AFI may answer with a Type B Answer To Request (*ATQB*).
- *SLOT-MARKER*: During anti-collision, the reader may send up to $N - 1$ *SLOT-MARKER* commands to indicate each time slot available for the tokens' *ATQB* responses. The commands can be sent after the end of an received *ATQB* message to mark the start of the next slot or earlier if no *ATQB* is received and it is known that the slot will be empty.
- *ATTRIB*: The *ATTRIB* command is used by the reader to select a single token. Upon receiving an *ATTRIB* command containing its identifier, a token enters the ACTIVE state where it only responds to commands defined in ISO/IEC 14443-4 that includes the Card Identifier (CID) assigned to it in the *ATTRIB* command parameters.
- *HLTB*: The Type B Halt (*HLTB*) puts the token into the HALT state.

When the token enters the interrogation zone of the reader, it powers up and enters the IDLE state. The anti-collision procedure, based on a dynamic slotted ALOHA algorithm, is started when the *REQB* command is transmitted. The token checks to see whether it has an application that matches the received AFI parameter and if this is the case it calculates the number of anti-collision slots N from the parameters in the *REQB* command. If $N = 1$, the token responds immediately with its *ATQB* response. Alternatively, the token is put in the READY REQUESTED state and randomly calculates a slot in which to send its response. In a probabilistic system, which does not use time slots, a token responds only if its randomly chosen slot is equal to 1. If it chooses any other slot, it returns to the IDLE state and waits for the anti-collision procedure to start again. In a pseudo-deterministic system that uses multiple slots, the tokens respond within the time slot corresponding to its randomly chosen slot. Once a token has responded with an *ATQB*, it is in the READY DECLARED state. The *ATQB* response contains information that the reader can use to identify and select the token. Once the reader has received a collision-free *ATQB* from the card it wants to select, it uses the *ATTRIB* command to place the chosen token into the ACTIVE state.

13.4.1.4 Part 4—Transmission Protocol

The final part of the standard specifies a half-duplex block transmission protocol for communication between the reader and the token. In Type A tokens, additional setup parameters need to be exchanged between the token and the reader to configure the protocol, such as frame size and card identifier. In Type B tokens, this is not necessary as these parameters have already been exchanged in the *ATQB* response and the *ATTRIB* command. When the Type A token is selected, the select acknowledgment will contain information about whether the token implements a proprietary protocol, or whether it supports ISO 14443-4. If a protocol adhering to ISO 14443-4 is available, the reader will transmit a Request for Answer To Select (*RATS*) command to which the token replies with an Answer To Select (*ATS*). This is followed by Protocol and Parameter Selection (*PPS*), if supported, that allows for the change of data rate between the token and the reader. The transmission protocol itself allows for the transmission of an Application Data Unit (APDU) that can contain any required data. The structure of the protocol is based on the $T = 1$ protocol specified in ISO 7816-3 for contact cards and is therefore often referred to as $T = CL$. This simplifies integration of contactless applications into smart card operating systems that are already available, especially in dual-interface tokens.

13.4.2 ISO 15693

ISO 15693, titled *Identification Cards—Contactless Integrated Circuit Cards—Vicinity Cards*, is most often implemented in systems using memory tokens for tracking or simple identification. Part 1 of the standard is very similar to the corresponding part in ISO 14443, so this section only discusses Parts 2 and 3.

13.4.2.1 Part 2—Air Interface and Initialization

This part of the standard describes the signal characteristics of the radio interface between the token and the reader, which allows for both power transfer and bidirectional communication. The token's power is provided by an alternating magnetic field at a frequency of 13.56 MHz and the reader must ensure that the magnetic field is within the range 115 mA/m $\leq H \leq$ 7.5 A/m. The operational range for systems using ISO 15693 can extend up to 1 m.

ISO 15693-2 defines both 'long distance' and 'fast' communication modes. In the 'fast' mode, the forward channel uses 1 of 4 pulse-position coding that is modulated onto the 13.56 MHz carrier with 100% ASK. One symbol comprises 8 time slots each of duration 9.44 μs and the modulation pulse can only be transmitted at an uneven time slot. The value n of the symbol can be determined by pulse slot $= (2 \cdot n) + 1$ and can be 0, 1, 2, or 3. One symbol takes 75.53 μs $= 8 \times 9.44$ μs to transmit, but each symbol can convey two bits of data, so the data rate is 26.48 kbps. For the 'long distance' forward channel data 1 of 256 pulse-position coding is used, which is then 10% ASK modulated onto the 13.56 MHz carrier. One symbol now comprises of 512 time slots and takes 4.833 ms to transmit. Since the symbol can have any value between 0 and 255, it can represent 8 bits of data so the effective data rate is 1.65 kbps. The backward channel uses Manchester coding, which can either be ASK modulated onto a 423 kHz sub-carrier or FSK modulated with a 423/485 kHz sub-carrier before being load modulated onto the 13.56 MHz carrier. For the 'long distance' mode, the data rate is 6.62 kbps, and for the 'fast' mode, the data rate is 26.48 kbps. Data is transmitted in frames marked by start-of-frame and end-of-frame symbols.

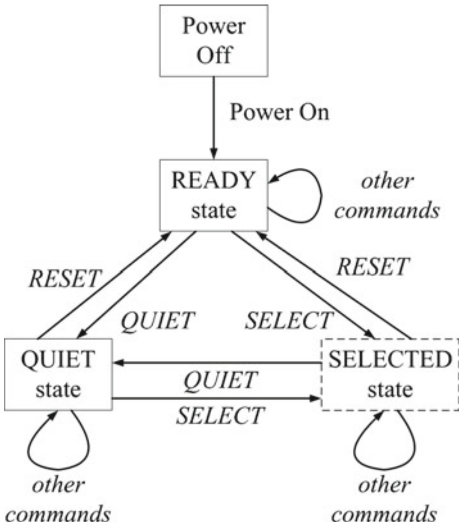
13.4.2.2 Part 3—Anti-collision and Transmission Protocol

The final part of the standard provides details on the anti-collision procedure, token initialisation, and possible commands. It should be noted that a large section of the guidelines given in this part is optional. As in ISO 14443 the token acts like a state machine during initialisation. A possible state machine diagram, given as an example in the standard, is shown in Fig. 13.12. The standard's transmission protocol specifies the format for command requests and responses, including a format for a number of optional commands such as *SELECT*, *RESET*, *READ* and *WRITE*. The standard only specifies two mandatory commands:

- *INVENTORY*: The token shall perform anti-collision when receiving the *INVENTORY* request.
- *STAY-QUIET*: If the token receives the *STAY-QUIET* request, it shall enter the QUIET state.

When the token enters the interrogation zone of the reader, it enters into the READY state. The reader will poll for tokens by transmitting an *INVENTORY* request. If a token is present, it will participate in the anti-collision procedure. The token compares the mask value from the *INVENTORY* request, varying in length from 0 to 8 bytes, with the corresponding least significant bits in its 64-bit unique

Fig. 13.12 ISO 15693: possible token state machine



identifier. If it is a match, the token will reply with its identifier during one of sixteen slots marked by the reader. It is assumed that the token does not have the necessary resources to randomly choose a slot. It therefore uses the four least significant bits not compared with the mask to determine the slot number. For example, if the 2 least significant bytes of the tokens UID is *4FAC* and the mask was *FAC*, then the token will respond in slot number 4. Once the reader has the token's unique ID, it can put the token in the QUIET state, or in the SELECTED state, if supported, where further commands can be issued.

13.4.3 ISO 18000

ISO 18000, titled *Information Technology AIDC Techniques—RFID for Item Management—Air Interface*, defines a generic structure for use in item management applications (Part 1), along with air interfaces for operation at 135 kHz (Part 2), 13.56 MHz (Part 3), 2.45 GHz (Part 4), 5.8 GHz (Part 5), 860–930 MHz (Part 6), and 433 MHz (Part 7). It is expected that all the parts of ISO 18000 will still be revised to include fixes and allow for the extra capabilities, such as active tokens and sensors [22]. This section only gives a brief overview of the standard in comparison with Part-2 of the ISO 14443 and ISO 15693 standards.

13.4.3.1 Part 3—Parameters for Air Interface Communications at 13.56 MHz

ISO 18000-3 defines a physical layer, collision management system, and protocol values, in accordance with ISO 18000-1, for RFID systems operating at 13.56 MHz. It specifies two modes of operation intended for use with different applications. Mode 1 is based on ISO 15693 with additional specifications to allows for item management applications and improved vendor compatibility. The reader to token data rate is 1.65, or 26.48 kbps. The token-to-reader data rate is 26.48 kbps. A protocol extension allows for data rates of 53 and 106 kbps. Mode 2 specifies a high-speed interface where the reader to tag data rate is 423.75 kbps.

13.5 Near-Field Communication

Near-Field Communication (NFC) is a contactless technology which enables electronic devices to communicate with each other by bringing them into proximity to a distance of several centimeters or less. NFC was initially established by Sony NXP (previously Philips Semiconductors) and Nokia in 2002, with the purpose of integrating RFID technology into mobile and embedded devices. NFC operates at the same frequency (13.56 MHz) and the radio interface shares several properties with existing HF RFID systems. Compared with RFID technology, NFC's bidirectional communication ability is ideal for establishing connections with other technologies by a simple touch, while RFID is usually used for a reader identifying or authenticating a tag which acts as a transponder. In addition, as a subset of HF RFID, NFC has the advantage of the short read range limitations of its radio frequency, which makes it a popular choice for intuitive communication between consumer devices, such as smartphones. Given its compatibility with existing systems, and offering new application possibilities, NFC has been widely deployed in our daily life. The following are some examples of the applications of NFC:

- *Ticketing*: Using our smartphone as a travel card, NFC works with most contactless smart cards and readers, which means it could be integrated into the transportation systems. For example, in 2008, German rail operator Deutsche Bahn launched an NFC-ticketing pilot program in which the travellers only needed to touch their phones to an NFC tag when boarding the train and then to another when getting off [23]. After that more and more countries choose to deploy NFC-enabled devices in public transportation. Among them, Iran trailed NFC-ticketing systems in 2012 [24].
- *Sharing*: two smartphones supporting NFC can share information with each other such as business cards. For example, Google announced NFC-based Android beam for sharing between phones in 2011 [25].
- *Service Discovery*: advertisers and marketers can use NFC chips in posters to promote their information, and people only need to touch their phones to these 'smart' posters to obtain the information.
- *Payment*: NFC works in a short range (less than 10 cm), which makes it a good choice for secure transactions such as contactless credit card payments. Examples include Google Wallet [26] and Apple Pay [27].

13.5.1 Standards—ISO 18092

ISO 18092, which is also referred to as NFCIP-1 (*Near-Field Communication Interface and Protocol*) or ECMA 340, specifies an RF interface and transmission protocol for communication between two inductively coupled devices operating at a frequency of 13.56 MHz. This standard allows for an active device, such as a mobile phone or PDA, to access RFID applications by acting as a reader, or a passive token, and it can also be used for short-range peer-to-peer communication. Although relatively new, NFCIP has strong support from industry. A NFCIP device can operate in three different ways, and it has more resources than a passive token, thereby allowing it to interface with its environment in a number of ways. The standard itself and the additional specifications for data exchange formats, record types, and compatible tokens are therefore quite comprehensive. This section only briefly discusses the different modes of operation and the sections of the standard corresponding to Part-2 of the

ISO 14443 and ISO 15693 standards. The reader is encouraged to read the ECMA 340 documentation [28] or visit the NFC Forum [29] for more details.

This standard defines ‘active’ and ‘passive’ communication modes between two entities, referred to as the target and the initiator. In active mode, both the initiator and the target generate a RF field. The initiator will start communication and transmit data by modulating its own carrier. Once it has finished transmitting, it will switch off the carrier and wait for a response. This is similar to two readers transmitting data to one another. The target then switches on its carrier and transmits a response. In the passive communication mode, only the initiator generates an RF field and starts the communication by modulating data on its own carrier. The target then responds to the initiator using a load modulation scheme. In this mode, one device acts like a reader while the other device emulates a passive token. Each device must ensure that it generates a magnetic field, at a frequency of 13.56 MHz, that is, within the range 1.5 A/m $\leq H \leq$ 7.5 A/m. The operational range for NFCIP systems is in the order of a few centimetres.

Both active and passive modes are defined for communication rates of 106, 212, and 424 kbps. The method for transmitting data at 106 kbps in passive mode is the same as for ISO 14443A. 106 kbps data transmission in active mode uses the same modulation scheme as the forward channel in ISO 14443A. For 212 and 424 kbps ‘passive’ and ‘active’ modes, both the forward and backward channel use Manchester code that is ASK modulated onto the 13.56 MHz carrier with a modulation index of 8–30% [7]. It should be noted that the backward channel does not use sub-carrier modulation. Given that NFCIP technology was initially developed by Nokia, NXP and Sony, the lower layer communication is compatible with NXP Mifare (106 kbps) and Sony FeliCa (212/424 kbps) products.

13.5.2 NFC Forum Specifications

To promote implementation and standardisation of NFC technology, the NFC Forum [29] is created by NXP, Sony, and Nokia at 2004, currently owning over 190 member companies. The NFC Forum has developed various technical specifications which can help achieve full interoperability between existing technologies and devices to enable new services. There are five types of specifications in total.

- *Protocol Technical Specifications*: This type of specifications deals with communication protocol between NFC devices. It includes Logical Link Control Protocol (LLCP) Technical Specification, Digital Protocol Technical Specification, Activity Technical Specification, Simple NFC Data Exchange Format (NDEF) Exchange Protocol Specification, and Analog Technical Specification. The details of the functionalities of each specifications can be found at [29].
- *Data Exchange Specifications*: This type of specification consists of the NDEF Technical Specification, which specifies a common data format for NFC Forum-compliant devices and tokens.

- *NFC Forum Tag-Type Technical Specifications*: This type of specification defines types of NFC tokens, which can be divided into four tag types. The four different tag operation specifications provide the technical information that is needed to implement the NFC devices. All the four types of tag are based on existing contactless products. Both Type 1 tag and Type 2 tag are based on ISO/IEC 1443A, and they can be read and rewrite capable. The memory availability of Type 1 and Type 2 tags is from 96 bytes to 2K bytes, and from 48 bytes to 2K bytes, respectively. The Type 3 tag, also known as FeliCa, is based on the Japanese Industrial Standard X6319-4. The theoretical memory availability is up to 1M bytes. Type 4 tags are fully compatible with the ISO/IEC 14443 standard series. Their memory availability is up to 32K bytes.
- *Record-Type Definition Technical Specifications*: This type of specification provides the technical specifications for Record-Type Definitions (RTD) and four specific RTDs including Text, URL, smart poster, and generic control. These specifications specify the format and rules for building standard record types and allow users to create their own applications based on these NFC Forum specifications. Readers can refer to [29] for the details.
- *Reference Application Technical Specifications*: This type of specification specifies the reference applications using NFC. Examples given in the NFC Forum include Connection Handover Technical Specification and Personal Health Device Communication Technical Specification. The former enables developers to define information needed for the connection setup to be carried in NFC Data Exchange Format messages. The latter supports an interoperable data transport for personal health devices conforming to the ISO/IEEE Standard 11073-20601 Optimized Exchange Protocol and NFC Forum specifications.

13.5.3 Mobile NFC Architecture

NFC-enabled mobile devices all have the three required components of the basic NFC architecture. The three basic components are the application execution environment (the normal execution and storage area of the device), the NFC module (radio frontend), and the secure element (trusted, secure execution and storage area). Figure 13.13 shows an example of a secure element and NFC module, together with an integrated antenna, in a phone circuit layout.

13.5.3.1 NFC Module

The NFC module handles the underlying communication functions. As introduced in Sect. 13.5.1, there are active and passive communication modes for NFC. In active mode, both initiator and target devices communicate by alternately generating their own fields, while in passive mode, the target device draws power through electromagnetic induction from the carrier field provided by the initiator device. Both active

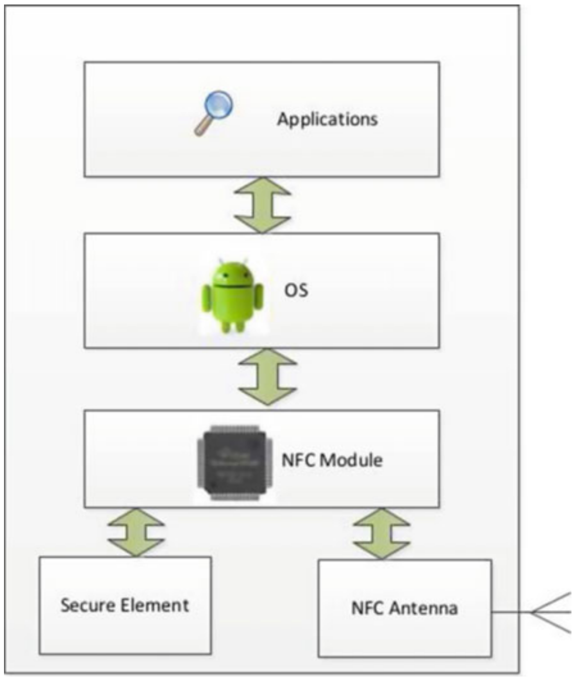
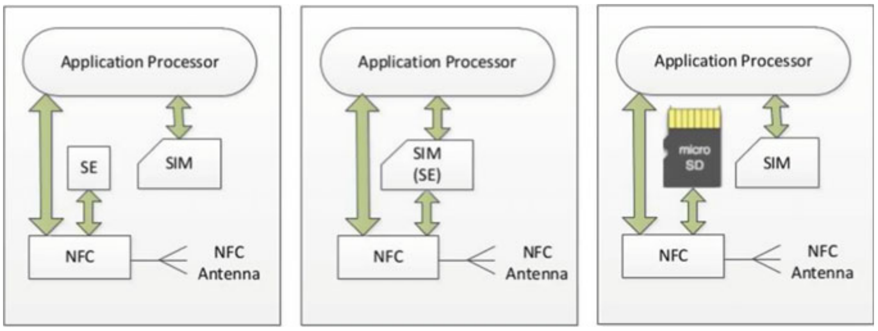


Fig. 13.13 Mobile NFC architecture

and passive modes are defined for communication rates of 106, 212, or 424 kbps. NFC-enabled devices can work in three operation modes: card emulation mode, peer-to-peer mode, and reader/writer Mode.

- *Card Emulation Mode*: In this mode, NFC-enabled devices act like smart cards, allowing users to perform transactions such as ticketing and purchases.
- *Peer-to-Peer Mode*: This mode enables two NFC-enabled devices to interact with each other such as exchanging information and sharing files.
- *Reader/Writer Mode*: In this mode, NFC-enabled devices can read information stored on inexpensive NFC tags embedded in smart posters and displays.

The NFC module has multiple interfaces to the Secure Element (SE), including ETSI 613 SWP (Single Wire Protocol) [30] and ETSI 622 HCI (Host Controller Interface) [31]. SWP is an interface between the Contactless Frontend (CLF) and the UICC (SIM card chip), only running on data link layer for communication. The CLF acts as a master and the UICC as a slave which can be powered by the CLF. HCI provides the network, transport, and session layers of the logical interface which runs over SWP. It supports multiple SEs controlled by a single NFC controller. An SE running on an application is referred to as a host while the NFC controller is referred to as a host controller.



(a) Embedded SE. (b) SIM as SE. (c) MicroSD SE

Fig. 13.14 Three basic SE architectures

13.5.3.2 Secure Element (SE)

The SE is for NFC-enabled mobile devices to perform secure transactions and store sensitive data in a trusted environment. It provides a secure means to establish trust between service provider and the device. Currently, SEs are commonly deployed in mobile devices with three different architectures, or a hybrid architecture that is a combination of the basic three architectures.

- *Embedded SE*: In this architecture as shown in Fig. 13.14a, the SE exists as an independent embedded hardware module (i.e., a stand-alone integrated chip) and is built into the phone. It could be seen as a mobile Trusted Platform Module (TPM) and, at the time of writing, is only used as card emulation.
- *SIM SE*: In this architecture, the SE is implemented within the existing SIM card (see Fig. 13.14b). There is no extra hardware. There are two variations: DIF-SIM where all functionality is on SIM with antenna in phone, and SIM-Flex where all functionality is on SIM with attached antenna.
- *micro-SD SE*: In this architecture, the SE is built in a removable memory component such as a micro-SD memory slot as shown in Fig. 13.14c. It can be added to any handset with an SD slot.

Although the SE is for providing secure NFC transactions, there are also some problems that need to be addressed. First, only card emulation mode really uses the SE, i.e., if the phone is acting as token, then the application is in the SE but if acting like a reader or in peer-to-peer mode, the SE is mostly not involved. Second, there is no standard security specifications, i.e., security is application specific. Finally, the SE memory is limited, since it is essentially a smart card. In addition, the card emulations are meant to run in an attack/tamper-resistant SE chip, and there are business/practical problems for service providers to get access to SEs, which inhibits services. To solve these problems, a solution first offered by BlackBerry and later by Android is Host Card Emulation (HCE).

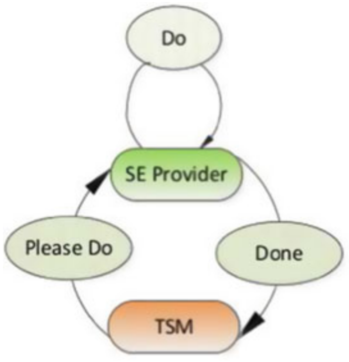
HCE is effectively a means to have a software SE running on the main phone processor, without the need for a physical secure element embedded as hardware in the phone. Instead of putting the SE inside the mobile device, HCE enables the SE to be placed in the phone application area or in a remote and hosted cloud environment and provides exact virtual representation of electronic identity cards using only software. In particular, with HCE, mobile applications can run on supported operating systems which offer payment card and access card solutions independently of third parties.

The term 'Host Card Emulation' was first introduced by Doug Yeager and Ted Fifelski, the founders of SimplyTapp, Inc., in 2012 [32] to describe the ability to provide a communication channel between a contactless payments terminal and a remotely hosted secure element which contains financial payment card data, allowing financial transactions to be conducted at a Point of Sale (POS) terminal. The first-known mobile handset to support HCE was the BlackBerry Bold 9900. With the release of Android 4.4, Google introduced a platform for secure NFC-based transactions through HCE, for payments, loyalty programs, card access, transit passes, and other custom services [33]. After the adoption by Google, HCE has also obtained support from many other big parties such as Visa, MasterCard and Microsoft. For normal users, however, HCE on Android is only available for application development when using an aftermarket Android release, such as *cyanogenmod* [34].

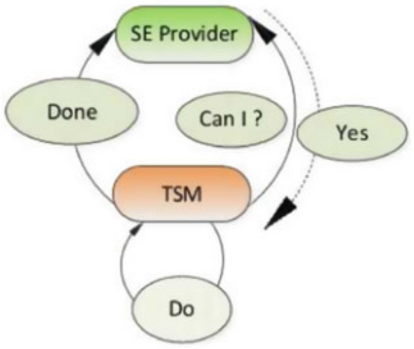
13.5.4 Basic Deployment Modes

NFC applications need to be securely placed into the secure element of devices OTA (Over the Air). This is different from a smart card. For example, the detailed information will be put onto the card by the smart card provider during the card personalisation once the bank arranges a bank card for a specific person. However, if a person buys a phone and then wishes to use it to pay, then the bank needs to put their application onto the phone while the phone is not under their control. The same trusted methods to personalise smart cards are therefore not available for NFC devices. For NFC-based applications, there are three different deployment modes [35].

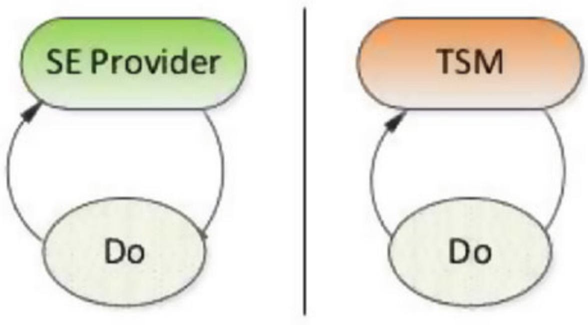
- *Simple Mode*: as shown in Fig. 13.15a, it is an issuer-centric model, where the Trusted Service Manager (TSM) requests the SE provider to perform a Card Content Management (CCM) operation and return the execution result to the TSM.
- *Delegated Mode*: in this mode, CCM is delegated to the TSM. Each operation requires preauthorisation from the SE provider. The execution result of the operation is optionally sent to the SE provider.
- *Dual Mode*: in this mode shown in Fig. 13.15c, CCM is fully delegated to the TSM on a dedicated area of the SE. Dual mode is characterised by the presence of at least two security domains with the authorised management privilege in the secure element.



(a) Simple Mode.



(b) Delegated Mode.



(c) Dual Mode

Fig. 13.15 Basic deployment modes [35]

13.5.5 NFC Security

Although NFC helps to enable various new services, it is faced with serious security issues. It is possible to unlock the SE in the embedded SE architecture mode. In particular, NFC is actually a nice attack platform for relay attacks, especially with HCE since an HCE can run any user application and does not need to be unlocked to be used for user-generated applications. NFC allows an attacker to have a programmable card for cloning applications and act as a proxy reader and proxy token in relay. In the following sections, a relay attack [36] and a cloning attack are illustrated that used NFC-enabled mobile devices as an attack platform.

13.5.5.1 Relay Attack

In a normal NFC communication system, two devices are in close physical proximity up to 10 cm. However, in a relay attack, the communication can be relayed

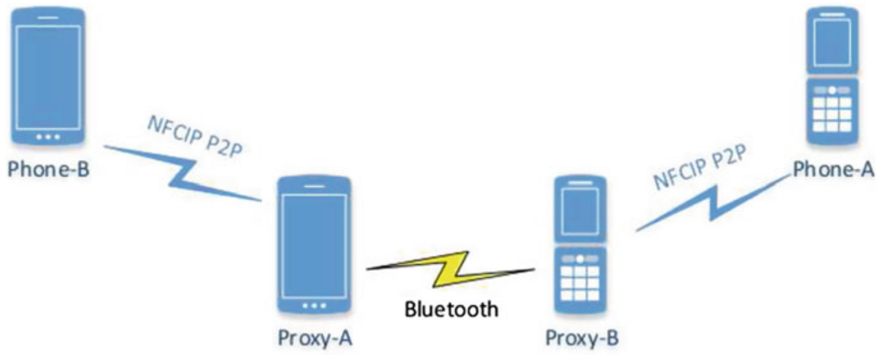


Fig. 13.16 Relay attack in NFC-enabled mobile devices communication [36]

over an extended distance by placing a proxy device within communication range of each legitimate participant and then forwarding the communication using another communication channel. The two legitimate participants receive valid transmissions from each other and therefore assume that they are in close physical proximity communicating with each other.

Figure 13.16 shows the relay attack against two NFC-enabled mobile phones operating in peer-to-peer mode and participating in a legitimate transaction demonstrated by Francis et al. [36]. The attack functionality can be implemented using only software which is written into the mobile devices Proxy-A and Proxy-B via publicly available APIs in a standard Mobile Information Device Profile (MIDP) using JSR 118 API [37]. In the presented experiment, Phone-A and Phone-B communicate with each other with the mobile-based proxy platforms, i.e., Proxy-A and Proxy-B. Proxy-B acts as the server and Proxy-A acts as the client, both of them establishing the relay channel using Bluetooth. Phone-A sends a message which is received by Proxy-B and relayed to Proxy-A over the Bluetooth data bearer. Then, Proxy-A transfers the payload onto Phone-B as well as relays the response from Phone-B to Proxy-B. Finally, Proxy-B transmits the response to Phone-A. In effect, Phone-A is made to believe that the response message is originating from Proxy-B while actually it is from Phone-B. Similarly, Phone-B is made to believe that the message is from Proxy-A which is actually from Phone-A. Fortunately, the authors in [36] proposed a countermeasure against the relay attack by using location information. By using NFC-enabled devices, it is also possible to relay communication between a normal contactless token and reader [38].

13.5.5.2 Cloning Attack

The first-generation contactless cards have rudimentary security mechanism, i.e., card is authenticated based on static data. In this case, the attacker can develop a mobile pick pocketing tool running on a NFC-enabled mobile phone and read data off the card. Then, the attacker can clone the card and use payment-reserved AID (Application ID) to communicate with a POS for a payment transaction. As a practical example, Roland and Langer [39] presented a cloning attack on EMV contactless payment cards where the adversary can create functional clones of a card that contains necessary credit data as well as preplayed authorisation codes. The card clones can then be used to perform a limited number of EMV Mag-Stripe transactions at any EMV contactless payment terminal.

13.6 Conclusion

Contactless tokens act as an electronic credential, interacting with the rest of the system on behalf of its owner. Contactless operation can increase productivity and convenience, while it offers additional benefits such as reduced maintenance cost and extended product lifetime. A choice of industry standards and tokens also allows this technology to be tailored to many applications. As a result, applications for contactless tokens have extended beyond access control and fare collection, with contactless tokens starting to replace, or supplement, established technologies such as paper tickets, barcodes, and magnetic stripe cards. This trend is expected to continue, especially in the identification and payment sectors, and with the introduction of NFC-enabled devices, the immediate future of contactless technology looks promising.

It is recommended that anybody wishing to learn more about contactless technology should consult literature by the Smart Card Alliance [11] and the '*RFID Handbook*' by Finkenzeller [18]. A number of open source RFID projects also provide hardware and software thatcan facilitate better understanding by means of practical experimentation [40–42].

References

1. R. Das. *RAIN RFID 2015–2020: Market size, growth opportunities and trends*. IDTechEx, 2015.
2. H. Stockman. *Communication by Means of Reflected Power*. Proceedings of the IRE, pp 1196–1204, October, 1948.
3. *The History of RFID Technology*. RFID Journal, December, 2006. http://www.rfidjournal.com/article/articleview/1338/1/129/.
4. Smart Card Alliance. *Contactless Technology for Secure Physical Access: Technology and Standards Choices.* Publication No. ID-02002, October, 2001.
5. ISO/IEC 14443. *Identification cards – Contactless integrated circuit cards – Proximity cards.*, 2011.
6. ISO/IEC 15693. *Identification cards – Contactless integrated circuit cards – Vicinity cards.*, 2009.
7. ISO/IEC 18092 (ECMA-340). *Information technology–Telecommunications and information exchange between systems–Near Field Communication – Interface and Protocol (NFCIP-1)*, 2013.

8. 2014 FIFA World Cup with RFID Ticketing Presents Cautionary Tale. June, 2014. https://www.tsl.com/2014/06/2014-fifa-world-cup-rfid-ticketing-presents-cautionary-tale/.
9. Federal Information Processing Standards. *Publication 201-1: Personal Identity Verification (PIV) of Federal Employees and Contractors*. March, 2006.
10. Federal Information Processing Standards. *Publication 201-2: Personal Identity Verification (PIV) of Federal Employees and Contractors*. August, 2013.
11. Smart Card Alliance. http://www.smartcardalliance.org/.
12. International Civil Aviation Organization (ICAO). *Document 9303 Machine Readable Travel Documents (MRTD). Part I: Machine Readable Passports, Seventh Edition*, 2015.
13. ISO/IEC 7501 *Identification cards – Machine readable travel documents*, 2008.
14. Bundesamt für Sicherheit in der Informationstechnik. *Advanced Security Mechanisms for Machine Readable Travel Documents' Extended Access Control (EAC).* Technical Guideline TR-03110, September, 2007.
15. T.S. Heydt-Benjamin, D.V. Bailey, K. Fu, A. Juels and T. O'Hare. *Vulnerabilities in first-generation RFID-enabled credit cards.* Technical report, University of Massachusetts Amherst, October 2006.
16. EMVCo. *EMV Contactless Communication Protocol Specification v2.5*. June, 2015.
17. EMV Integrated Circuit Card Specifications for Payment Systems-Book 1: Application Independent ICC to Terminal Interface Requirements. v4.3, November, 2011. https://www.emvco.com/specifications.aspx?id=223.
18. K. Finkenzeller, *RFID Handbook: Radio-frequency identification fundamentals and applications*, Wiley, 1999.
19. J.G. Proakis. *Digital Communications*, 5th Edition, McGraw-Hill, 2007.
20. J.G. Proakis and M. Salehi. *Communication Systems Engineering*, 2rd Edition, Prentice-Hall, 2002.
21. ISO/IEC 18000. *ISO/IEC 18000 Information Technology AIDC Techniques-RFID for Item Management – Air Interface*, 2010.
22. Institute for Prospective Technological Studies. *RFID Technologies: Emerging Issues, Challenges and Policy Options*, Technical Report EUR 22770 EN, 2007.
23. Marcus Gemeinder. *Touch & Travel C NFC based automatic fare collection using a passive infrastructure*. NFC Forum: Transport and City Life focus group, 2008.
24. Irancell demonstrates NFC payments and ticketing. http://www.nfcworld.com/2012/01/13/312430/irancell-demonstrates-nfc-payments-and-ticketing/, 2012.
25. Google announces NFC-based Android Beam for sharing between phones (video). http://www.engadget.com/2011/10/18/google-announces-nfc-based-android-beam-for-sharing-between-phon/, 2011.
26. Google Wallet. https://en.wikipedia.org/wiki/Google_Wallet.
27. Apple Pay. https://en.wikipedia.org/wiki/Apple_Pay.
28. ECMA-340 *Near Field Communication Interface and Protocol (NFCIP-1)* 3nd Edition, June, 2013. http://www.ecma-international.org/publications/standards/Ecma-340.htm.
29. NFC Forum. http://www.nfc-forum.org/.
30. ETSI TS 102 613-V11.0.0. *Smart cards: UICC– contactless Front-end (CLF) Interface; Part1: Physical and data link layer characteristics.* September, 2012.
31. ETSI TS 102 622-V11.1.0 *Smart cards: UICC– Contactless Front-end (CLF) Interface; Host Controller Interface (HCI).* October, 2012.
32. SimplyTapp proposes secure elements in the cloud. http://www.nfcworld.com/2012/09/19/317966/simplytapp-proposes-secure-elements-in-the-cloud/, 2012.
33. Host-based Card Emulation. https://developer.android.com/guide/topics/connectivity/nfc/hce.html.
34. CyanogenMod. https://en.wikipedia.org/wiki/CyanogenMod.
35. GlobalPlatforms Proposition for NFC Mobile: Secure Element Management and Messaging. Online whitepaper, GlobalPlatform, April, 2009.
36. Lishoy Francis, Gerhard Hancke, Keith Mayes and Konstantinos Markantonakis. *Practical NFC Peer-to-Peer Relay Attack Using Mobile Phones*. In: Radio Frequency Identification: Security and Privacy Issues, pages 35–49, LNCS, Springer, 2010.

37. Sun Microsystems, JSR-000118 Mobile Information Device Profile 2.0, https://jcp.org/aboutJava/communityprocess/final/jsr118/index.html, 2010.
38. Lishoy Francis, Gerhard Hancke, Keith Mayes and Konstantinos Markantonakis. *Practical Relay Attack on contactless Transactions by Using NFC Mobile Phones*. In: Journal of Radio Frequency Identification System Security RFIDsec'12 Asia Workshop Proceedings, pages 21–32, IOS Press, 2013.
39. Michael Roland and Josef Langer, *Cloning Credit Cards: A Combined Pre-play and Downgrade Attack on EMV Contactless*. In 7th Usenix Workshop on Offensive Technologies, Washington, D.C., 2013.
40. OpenPCD Project. http://www.openpcd.org.
41. rfdump Project. http://www.rfdump.org/.
42. rfidiot Project. http://www.rfidiot.org/.

Chapter 14
ID Cards and Passports

Ingo Liersch

Abstract In this chapter, we discuss e-ID cards and e-Passports. A number of countries have introduced electronic identity documents, and some other countries are planning the introduction. The reasons generally are requirements for a more secure and automated identification and additional functions such as e-Government. In October 2006 the United States required biometric enabled electronic passports for the Visa Waiver countries. The European Commission set the implementation time frame so that all member states had to implement the facial image requirements by 2006. These decisions led to worldwide introductions of e-Passports. With mandating the Supplemental Access Control (SAC) protocol in 2014, the European Union meanwhile introduced the third generation of electronic passports.

Keywords ID Cards · e-ID · e-Passport · Chip · ICAO · Security · Basic Access Control (BAC) · Extended Access Control (EAC) · Password Authenticated Connection Establishment (PACE)

14.1 Introduction

Security is a top priority in an age in which people fear terrorism and crime, fraud threatens national economies, while data and intellectual theft endanger the success of companies. Subsequently, governments are driven by their citizens' security demands. The increase in daily air traffic and the technological armament of impostors lead to new secure means of identification: e-ID smart cards and e-Passports.

The introduction of e-IDs and e-Passports is discussed not only in Europe but also worldwide.

Personal identity documents confirm the identity of individual citizens, thus proving their legitimate residency within their homeland. Government offices, credit institutes and businesses ask for ID cards or passports as a means of unequivocally

I. Liersch (✉)
Director Product Marketing, Government Identification, Infineon Technologies AG, Neubiberg, Germany
e-mail: ingo.liersch@infineon.com

identifying the holder. Identification on the Internet, e.g. for business and government transactions, requires a valid means of identification which can be provided by e-ID documents. In addition, opening a bank account, conducting valid business transactions and drawing government benefits can become impossible without a trusted means of identification. Therefore, it is in the best interests of any government for its citizens' personal identification documents to be made as difficult as possible to counterfeit or misuse.

Modern national ID cards comply with the ISO/IEC 7816—series standards and other international norms for signatures and certificates.

14.2 ID Cards

In this chapter, we only discuss cards which serve optically and digitally as an ID card. What all these cards have in common is that they are issued by a governmental institution or agency for citizens of their country. In some countries, this institution is the Ministry of Interior, and in other countries, the police or local government offices.

National ID cards, e-ID, or Smart ID: every country has its own term for this kind of smart card.

14.2.1 Requirements and Constituents of Modern National ID Cards

Unlike Subscriber Identity Module (SIM) cards and banking cards, where the relevance of the digital function is dominant, all the features of the whole ID document are important as these documents also serve as visual identification. An accepted expression is that an ID document must be designed in a way that a police officer on a foggy night is able to recognise a person via the document and easily verify the genuineness of the document. Nevertheless, the trend of the ID card becoming a smart e-ID card can be watched all over the world.

14.2.1.1 Card Body Material and Design

ID cards and passports can be subjected to many kinds of stress, which is why they need to be as durable as possible in order to resist tampering, chemicals, varying temperatures, environmental conditions and UV radiation. In addition, the images and text must not fade significantly with age (required lifespan is up to 10 years). As a result, card manufacturers had to develop materials for the card body, providing the highest level of counterfeit resistance and durability. Specially developed foils

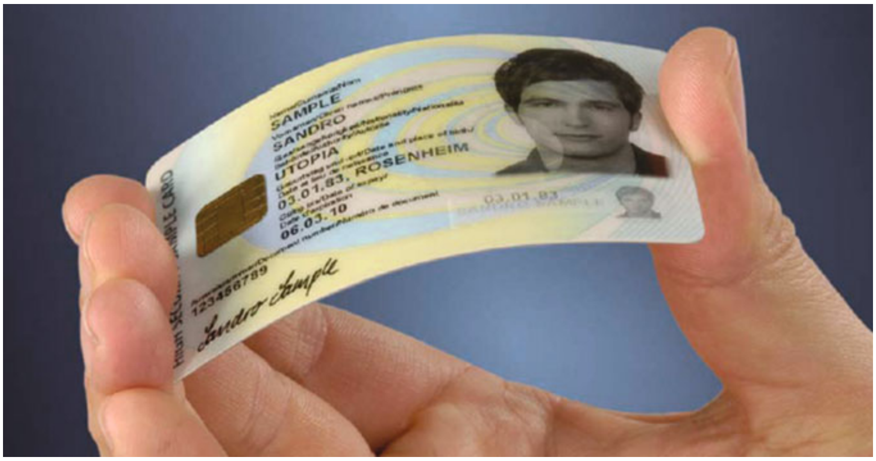


Fig. 14.1 Sample polycarbonate ID card (*Source* Giesecke & Devrient/Veridos)

for ID card production are only available for accredited security printers and card manufacturers (ID cards are always laminated multilayer cards).

Conventional materials such as PVC (polyvinyl chloride) used for credit cards do not meet the requirements for ID cards yet. A market-proven material is polycarbonate (PC) which is used in most ID projects with card life cycles between 5 and 10 years. A card made out of polycarbonate consists of 4–10 foils (depending on the card manufacturer) laminated together by high temperature and pressure. Polycarbonate cards are supposed to be very stable in terms of bending, torsion, UV light and temperature. Polycarbonate is also seen as the best material for laser personalization. However, the polished surfaces of ID cards or data pages made of PC tend to become dull within a short time due to the sensitivity of the material to scratching. Figure 14.1 shows a sample polycarbonate ID card.

A second material used for long life cards is polyester which is cheaper, more flexible and less scratch sensitive than polycarbonate. Polyester cards could be even more stable in terms of bending and torsion, but they could not reach the temperature resistance of polycarbonate cards. A possible polyester material used for cards is PETG (polyethylene terephthalate glycol-modified).

National ID cards are tailored to the individual country's history, culture and country-specific preferences for colours and patterns. A number of classical or unique security features are always combined in the design.

In Fig. 14.2, the design of the Egypt ID is shown. The background security printing shows the culturally very important pyramids of Egypt. The lettering is only in Arabic because it is only an ID document within Egypt or for travelling to friendly Arabian countries.

In Fig. 14.3, the Macao ID card is shown. A lotus is integrated in the background's design. The lettering is in Chinese and in English because both languages are used in the Special Administrative Regions (SARs) such as Hong Kong and Macao.

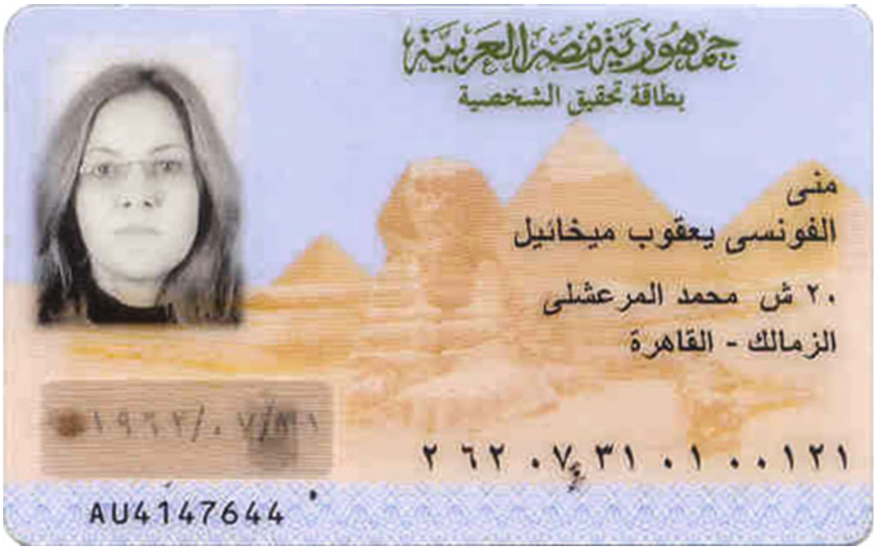


Fig. 14.2 Sample polycarbonate ID card: Egypt ID

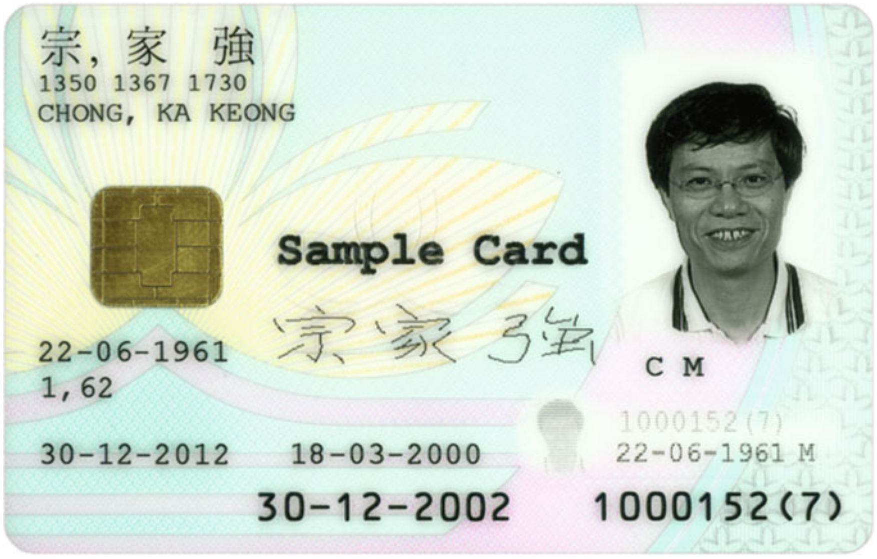


Fig. 14.3 Sample polycarbonate ID card: Macao ID

14.2.1.2 Card Body Security Features

Remark: Here can only be shown generic standard and mostly used security features. Features that are brand protected or owned by a company are not listed.

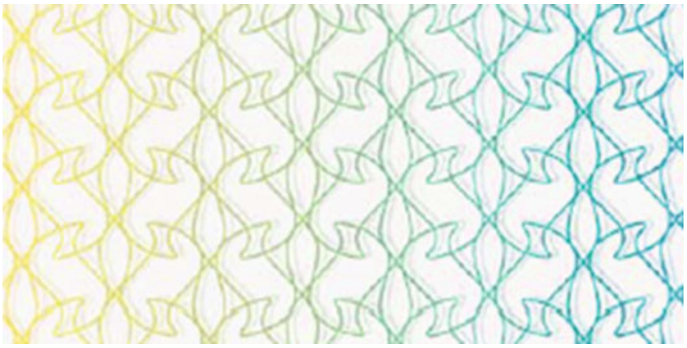


Fig. 14.4 Security background printing (*Source* Giesecke & Devrient/Veridos)

Security features can be divided into three verification levels (Industry Standard):

Verification level 1:	Features recognisable using human senses without the aid of tools;
Verification level 2:	Features recognisable with the aid of simple tools (usually UV lamp and magnifying glass); and
Verification level 3:	Features recognisable in a forensic laboratory or machine readable elements. This includes microprocessors.

A reasonable selection of security features covers all levels of verification. Therefore, security is not based on the purely number of features but on the intelligent combination of highly secure features which are easy to verify and cover several verification levels.

The following list shows generic security features for ID cards.

1. Verification level 1:

• This process allows relief structures to be worked with traditional guilloches resulting in a customer-specific design. The pallet of inks range from rainbow colours to those with fluorescent properties. This kind of design structures can be seen in principle on every banknote in every currency. Figure 14.4 shows security background printing.

• OVI (Optically Variable Inks): A high level of protection, particularly against colour-copied counterfeits, can be achieved using optically variable inks. Depending on the angle from which the feature is viewed, the ink changes colour, normally from green to blue to violet. Figure 14.5 shows an example using optically variable inks.

• MLI or CLI (Multiple or Changeable Laser Image): This is one of the most widely used anti-counterfeiting measurements as it protects the personal data of the ID document holder, data which counterfeiters find most attractive to attack. It consists of numerous, tiny horizontal lenses incorporated into the surface of the card with special laminating plates. Depending on the angle from which they are viewed, even in poor lighting conditions, the holder sees different bits of information, e.g. date of birth or a microphotograph. Due to the different pieces of information revealed

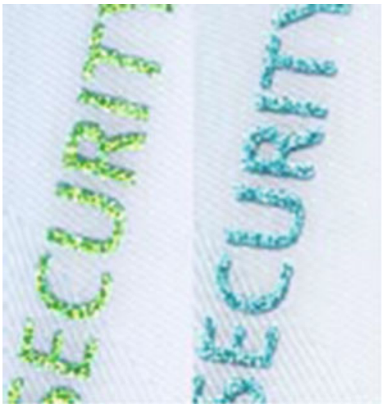


Fig. 14.5 OVI Optical Variable Ink (*Source* Giesecke & Devrient/Veridos)

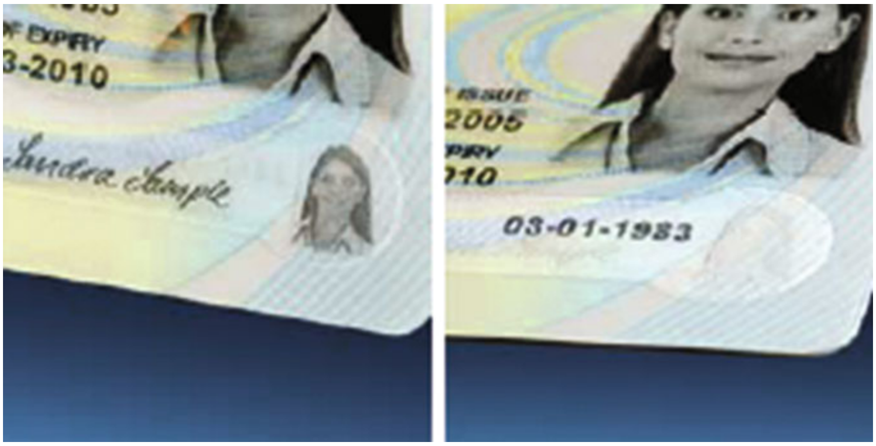


Fig. 14.6 MLI (multiple laser image) (*Source* Giesecke & Devrient/Veridos)

at various angles, this feature well protected from being copied or manipulated. Meanwhile, there are variations on the market combining this technology with other optically variable printed images or elements such as moiré patterns. Figure 14.6 shows an example of the multiple laser image.

2. Verification level 2:

• Microlettering: This feature, for example in the form of plain text, can only be seen through a magnifying glass. Attempts to copy microlettering usually fail to reproduce the feature faithfully. Figure 14.7 shows an example of microlettering.

Here, fluorescent inks are used, which are invisible under normal light and only reveal the hidden design under ultraviolet light. The challenge in creating these features is integrating them intelligently into the overall design. UV inks can be used for patterns, text or logos. Figure 14.8 shows an example of UV printing.

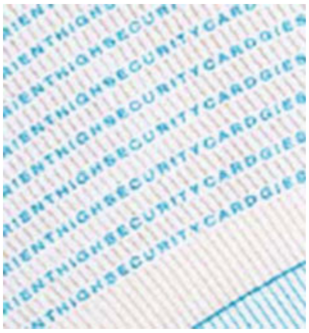


Fig. 14.7 Microlettering (*Source* Giesecke & Devrient/Veridos)

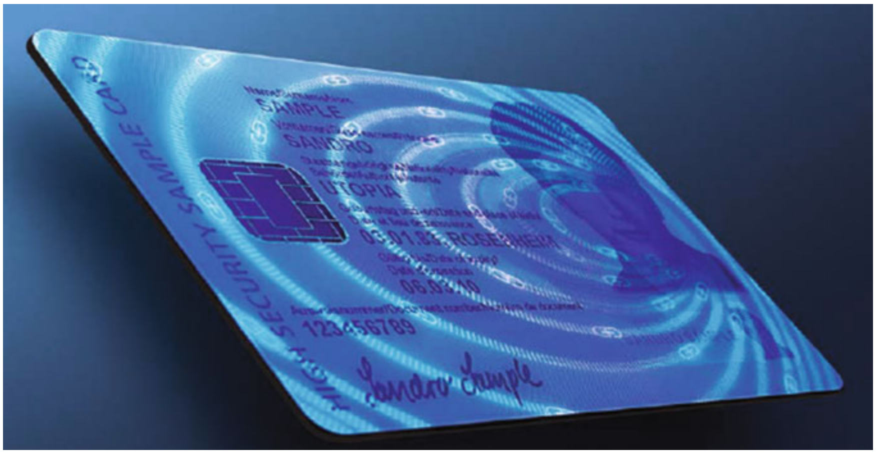


Fig. 14.8 UV printing (*Source* Giesecke & Devrient/Veridos)

• IR (infrared) printing: Inks or ink components that are only visible under infrared lighting can also help protect cards against counterfeiting. Since modern optical readers at border crossings already make use of infrared light, no additional equipment for verification is required.

3. Verification level 3:
Microprocessors offer numerous options for card administration, authentication, and data access control, as well as storage and verification of biometric features. This level can also comprise secret features known only to the manufacturer and the card issuer.

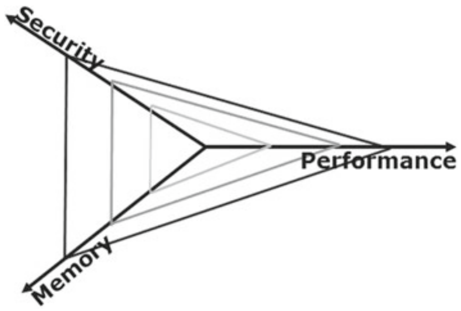


Fig. 14.9 Magic *triangle* for chip requirements

14.2.1.3 Chip Hardware for Electronic ID Cards

All manufacturers of security controllers for smart cards differentiate between platforms for credit cards, telecommunication and identification cards due to different requirements. Chip platforms for ID cards have to match the magic triangle of security, performance and memory (see Fig. 14.9 for an illustration).

This means that chip hardware is always a compromise between these requirements. For example, the highest security will be achieved at the expense of performance due to additional attack countermeasures and checks.

Security requirements are constantly increasing. Attacks on chips become more and more sophisticated.

One way on modern platforms to fight even currently unknown attacks is the double-core principle: two CPU's calculate the same and check each other's flawless operation. If the chip is being attacked, the induced errors can be detected, triggering an alarm state in the chip. An additional protection method is full data encryption—even in the CPU itself.

A general precondition for usage in ID projects is security certification according to Common Criteria or FIPS standards. Common Criteria EAL5+ is the minimum certification level for security controllers. High-level security controllers are meanwhile EAL6+ certified.

There is no typical NVM (non-volatile memory) size for e-ID platforms. The required memory completely depends on the country's individual profile for card applications. There is a definite trend towards multiapplication cards in ID schemes. Modern ID platforms can not only be used for identification (e.g. based on biometrics), but can also be used for online authentication, transport and payment or other applications.

Former controllers have been based on Read-Only Memory (ROM) and Electrically Erasable Programmable Read-Only Memory (EEPROM). For some time, these controllers are being replaced more and more by FLASH-based products. FLASH controllers reach the same or even higher security certification than EEPROM controllers. Due to a smaller cell size of flash, it is possible to realise security controllers with up to 1 MByte or more but still fitting in a smart card module. In the first ID card projects (from 2000 onwards), contact-based smart cards were issued.

Nowadays, a clear trend towards dual interface or contactless cards can be seen. There are several reasons for that: since 2006, many countries have been introducing electronic passports and installing the infrastructure for reading them. The same infrastructure can also be used for reading of dual interface or contactless ID cards. A second reason is the much faster transaction rate of a contactless interface. According ISO 14443, contactless transaction rates can be up to to 848 Kbit/s (standard e-Passport terminal) or even up to 6.8 MByte/s (VHBR = very high bit rate). VHBR leads to higher performance, especially in the context of biometric verification where large templates of biometric features such as fingerprints or facial images have to be read by an inspection terminal.

14.2.1.4 Card Operating Systems for ID Cards

The market is mainly split between native (proprietary) ISO card operating systems (e.g. compliant to ISO 7816 parts 1–4, 8, 9) and Java Cards (e.g. compliant to ISO 7816 parts 1–3 and corresponding Java Card and Global Platform specifications).

Both card operating systems have their specific advantages: both are secure and flexible. Additionally, almost every kind of application can be realised on either platform. As a result, the decision on the card operating system is often a political one.

For national e-ID cards, more and more Java Cards are deployed due to a higher flexibility in terms of loading or updating of applications.

14.2.1.5 ID Card Applications

Card applications are always country-specific. A clear trend is the implementation of the International Civil Aviation Organization (ICAO) travel application, which is described in the next section. As a result, ID cards can be used as an e-Passport for travelling (in countries accepting this kind of document). Generally, the following functionalities are realised in ID cards:

- online authentication functions;
- digital signature and encryption;
- storage of biometric information or on-card matching;
- storage of personal information; and
- e-voting.

14.2.2 International Standards for ID Cards

Standards applicable to national ID cards are shown in Table 14.1.

Table 14.1 International standards for ID cards

Standard	Comment
ISO/IEC 7816	ISO 7816 is an international standard related to electronic identification cards, especially smart cards
ISO/IEC 7810	Physical characteristics of ID cards
ISO/IEC 10373-3	Test methods of ID cards
ISO/IEC 18013 part 1–3	Drivers Licence
Java Card Specifications	If applicable
Global Platform Specifications	If applicable
ICAO Doc 9303 part 5 [5] (Technical Specifications are endorsed by ISO Standard 7501)	ICAO sets standards for Machine Readable (Official) Travel Documents such as passports and ID cards (MROTDs)

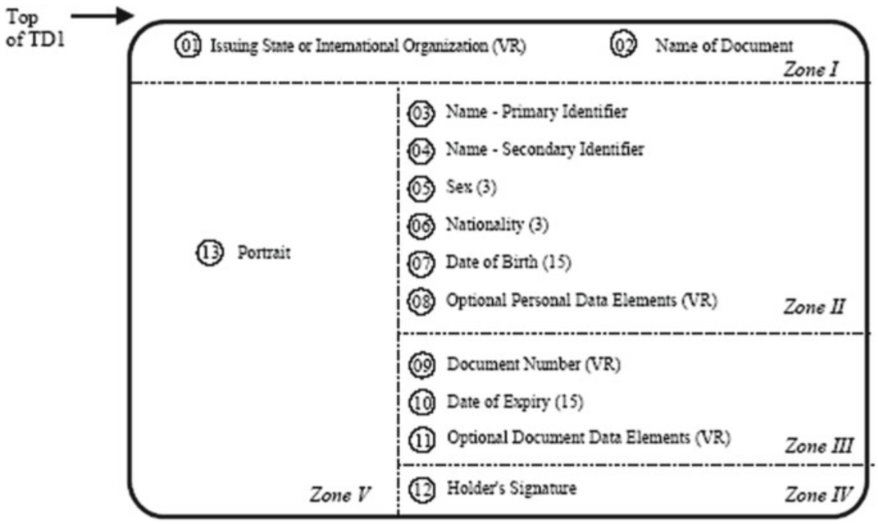


Fig. 14.10 General layout of the TD-1 MRTD according to ICAO

14.2.2.1 ICAO Requirements for ID Cards

In ICAO specification ICAO Doc9303 part 5, the layout of **M**achine **R**eadable **O**fficial **T**ravel **D**ocuments (MROTD) in ID-1 format is defined as shown in Figs. 14.10 and 14.11.

Zone I: Header
Zone II: Personal data elements
Zone III: Document data elements
Zone IV: Holder's signature

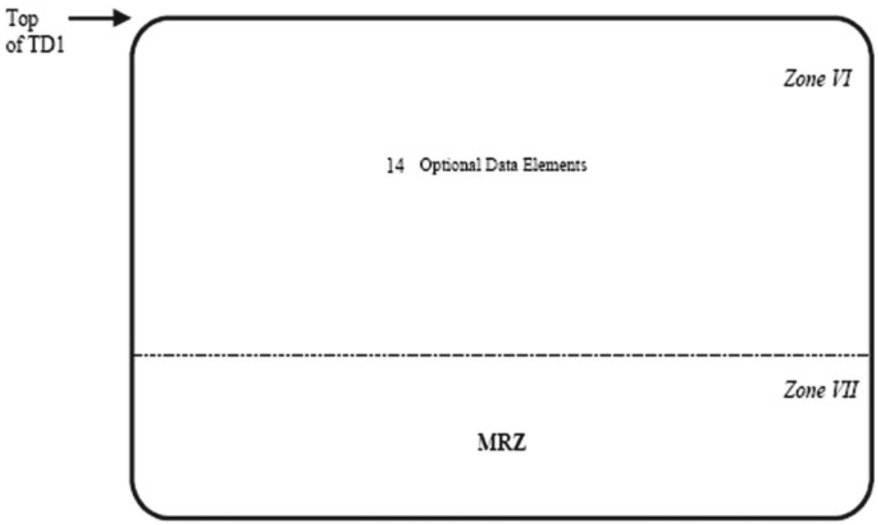
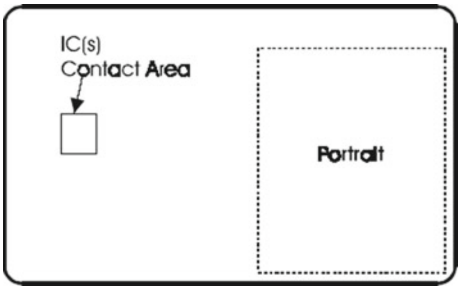


Fig. 14.11 General layout of the TD-1 MRTD according to ICAO (rear side)

Fig. 14.12 General layout of TD-1 MRTD with chip



Zone V: Identification feature
Zone VI: Optional data elements
Zone VII: MRZ

For ID cards containing a chip, the layout shown in Fig. 14.12 has to be applied:

ICAO also defines the so-called MRZ (Machine Readable Zone) for ID cards when they are used for travelling, as shown in Fig. 14.13. In European countries, citizens can generally travel with their ID cards and do not need to carry their passports. Although in the Schengen area, there are no border controls any longer, these cards must feature a MRZ.

Please note that passports have a MRZ comprising only two lines.

Besides the possibility to read the traveller's personal data in a fast and accurate way, the MRZ is used for Basic Access Control (BAC) or Password Authenticated Connection Establishment (PACE) (please see Sect. 14.3.4 "Security Protocols" for more details).

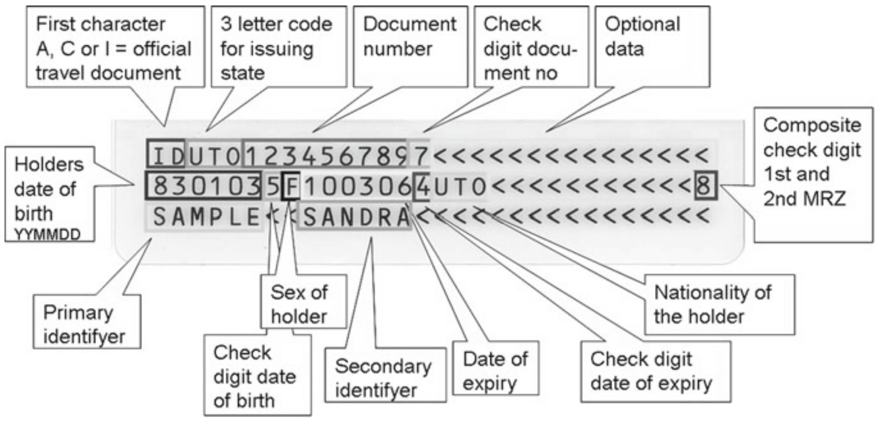

First character A, C or I = official travel document
3 letter code for issuing state
Document number
Check digit document no
Optional data
Composite check digit 1st and 2nd MRZ
Holders date of birth YYMMDD
IDUTO1234567897<<<<<<<<<<<<<<<
8301035F1003064UTO<<<<<<<<<<<8
SAMPLE<<SANDRA<<<<<<<<<<<<<<<<
Primary identifyer
Sex of holder
Check digit date of birth
Secondary identifyer
Date of expiry
Nationality of the holder
Check digit date of expiry

Fig. 14.13 Structure of MRZ for ID cards

14.2.3 Optical Personalisation of ID Cards

This section provides an introduction to available technologies for optical personalisation of plastic cards.

14.2.3.1 Thermo-Transfer Printing

By heating up individual elements within a print head, the so-called dyes evaporate from the donor ribbon. The dyes are transferred onto the card surface. The temperature of every single element defines the amount of colour to be transferred and also controls the brightness of the pixels. Figure 14.14 shows the thermo-transfer printing process.

The parameters to be adjusted are the temperature of the print head's elements and the combination of donor ribbon and card body. Slight changes of temperature can lead to a pale or blurred print image.

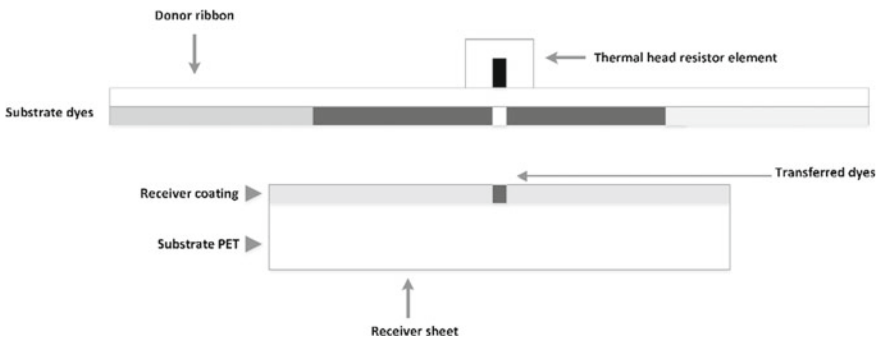


Fig. 14.14 Thermo-transfer printing process

General Features:

- consumables are the donor ribbons;
- thermal heads have a restricted life time (temperature stress);
- colour photographs possible; and
- PVC and polyester cards can be personalised.

Strengths of Thermo-Transfer Printing:

- small and cheap printers and
- colour photograph can be realised.

Weaknesses of Thermo-Transfer Printing:

- no polycarbonate material can be personalised;
- irregular printing around security features such as relief structures on the card surface;
- irregular printing around a chip module;
- no protection against abrasion and UV light;
- blank cards must be absolutely clean;
- no protection against fraud: data and pictures can easily be removed or changed (see example below);
- for the use of secure ID cards, an additional protective lamination foil on the surface is necessary; and
- high costs due to consumables: cleaning cards, cleaning rolls, colour ribbons, lamination foils, thermal heads, drive rolls etc.

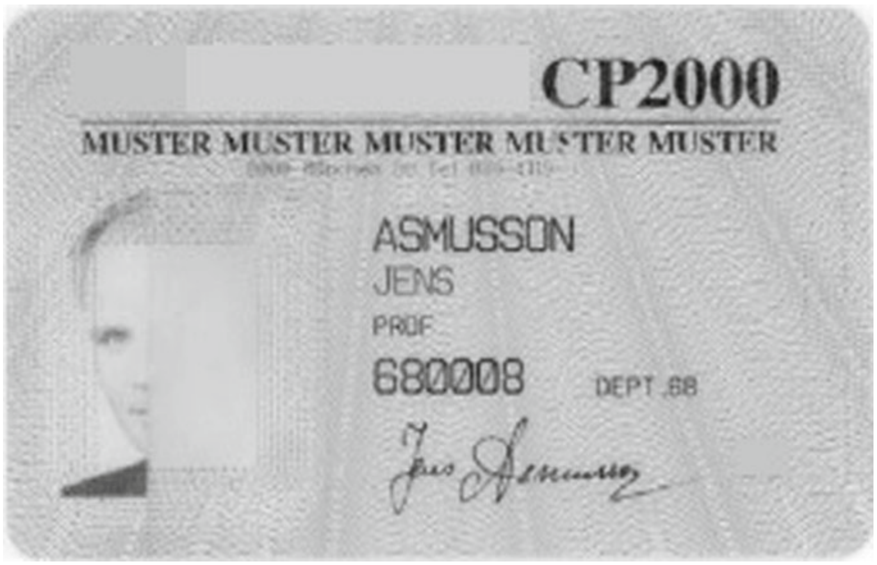


Fig. 14.15 Manipulation of a colour photograph personalised with thermo-transfer printing (*Source* Giesecke & Devrient/Veridos)

Figure 14.15 demonstrates the ease with which we can manipulate colour images: The image could be erased manually with nearly no traces left. A different photograph can be added afterwards by a counterfeiter.
Summary:
Thermo-transfer is a technology only suitable for small numbers of cards (e.g. access cards for a company) with little need for protection and short-to-medium lifetime. Thus, this technology cannot be recommended for national ID cards.

14.2.3.2 Thermo-Sublimation Printing

Thermo-sublimation printing is similar to the thermo-transfer process described above. In this process, special coloured wax is applied on the card surface. When exposed to very high temperatures (300–400 °C), the wax is sublimated (converted to a gaseous status) and then coated on the card. Due to the high temperature, the colour penetrates the material slightly and gives some protection against abrasion and UV light. In addition, very high print quality can be achieved. Nevertheless, a protecting foil is necessary for high security cards.
Positive Features of Thermo-Sublimation Printing:

- high printing quality;
- higher protection against abrasion than thermo-transfer technique;
- limited protection against fraud in combination with a protection foil; and
- colour photograph can be realised.

Negative Features:

- expensive printer;
- expensive consumables (even higher costs than with thermo-transfer);
- protection against fraud only together with additional foil.

Summary:
In terms of quality, it provides a better choice than thermo-transfer. Limited security is possible in combination with a protection layer.

14.2.3.3 Thermo-Retransfer Printing

Thermo-transfer printing has been further developed to thermo-retransfer printing. By heating up the print head, the colour will be transferred mirror-inverted by thermal transfer printing from a ribbon to a special retransfer film. The image is then transferred onto the card by heat roller. This is a lamination process under pressure and heat. The laminated film is about 30 μm thick.

By using this retransfer process, a high resolution is possible.
Strengths of Thermo-Retransfer Printing:

- independent from card surface (relief structures) or contact chip modules;
- over the edge printing; and
- polycarbonate cards can be personalised.

Weaknesses of Thermo-Retransfer Printing:

- expensive printers and
- high costs for consumables (additional retransfer film necessary, for one card one whole set of colour segments Cyan-Magenta-Yellow-Black (CMYK) is consumed).

Summary:
Security is comparable to thermo-sublimation with protection foil. As long life cards made of polycarbonate can be personalised, cards with a requirement for a long lifetime and medium-to-high security can be realised.

14.2.3.4 Laser Engraving

When using a laser engraving technique, a laser beam penetrates through the transparent cover foils through to the opaque inner foil. As a result, the information is engraved into every layer of this side of the card. As a prerequisite, the card body material must be designed for laser engraving. Figure 14.16 shows the laser engraving process. Engraved images are extremely durable, and they resist chemicals and withstand physical contact easily. Alteration is nearly impossible without destroying the card.

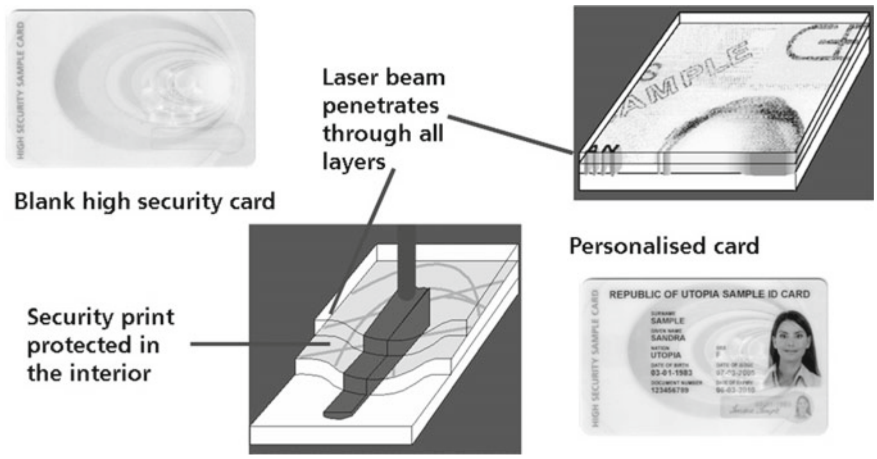

Laser beam penetrates through all layers
Blank high security card
Personalised card
Security print protected in the interior
REPUBLIC OF UTOPIA SAMPLE ID CARD
SAMPLE
SANDRA
UTOPIA
03-01-1983
123456789
HIGH SECURITY SAMPLE CARD

Fig. 14.16 Laser engraving process (*Source* Giesecke & Devrient/Veridos)

Strengths of Laser Engraving:

- highly secure personalisation technology;
- no consumables;
- no abrasion by card usage;
- no change by UV light;
- higher resolution than with colour printing; and
- lower total costs (no consumables).

Weaknesses of Laser Engraving:

- higher costs for the machinery and
- black and white photographs

Summary:
Laser engraving systems allow card issuers to permanently engrave photographs and other personal data on their cards. When comparing laser-engraved black-and-white images with colour images, it is widely accepted that laser engraving enhances the overall card security substantially.

14.2.4 Countries and Their ID Cards

A number of countries have introduced e-ID cards, and some other countries are planning the introduction. In this section, a few examples are given.

14.2.4.1 Finnish Electronic Identification (FINEID)

The FINEID shown in Fig. 14.17 was the first worldwide example of the combination of an ID card with an electronic signature card. The pilot phase started in 1998.

Features:

- issuance of cards since 2001;
- electronic signature card;
- not mandatory for citizens;
- includes no biometrics;
- available for foreigners after 6-month residence;
- contact interface; and
- optical personalisation technology: laser engraving.

14.2.4.2 Estonia ID

In Estonia, ID cards are issued to every citizen over the age of 16. An interesting feature is the authentication certificate containing the citizen's email address which can be used by government institutions or private persons (see Fig. 14.18).

Fig. 14.17 FINEID

Fig. 14.18 Estland ID

Features:

- since 2002;
- signature card;
- mandatory for all citizens (from 16);

- no biometrics;
- official email address assigned by government;
- contact interface; and
- optical personalisation technology: laser engraving.

14.2.4.3 German ID Card nPA (= Neuer Personalausweis)

Germany introduced a new electronic ID card in 2009, shown in Fig. 14.19. The card features following applications:

The card will feature the following applications:

- ICAO travel application including facial image and two fingerprints;
- online authentication application; and
- optional: qualified digital signature.

The card only has a contactless interface (RFID) like an electronic passport. The citizen can decide himself about the activation of the online authentication application and even about the storage of the two fingerprints.

Features:

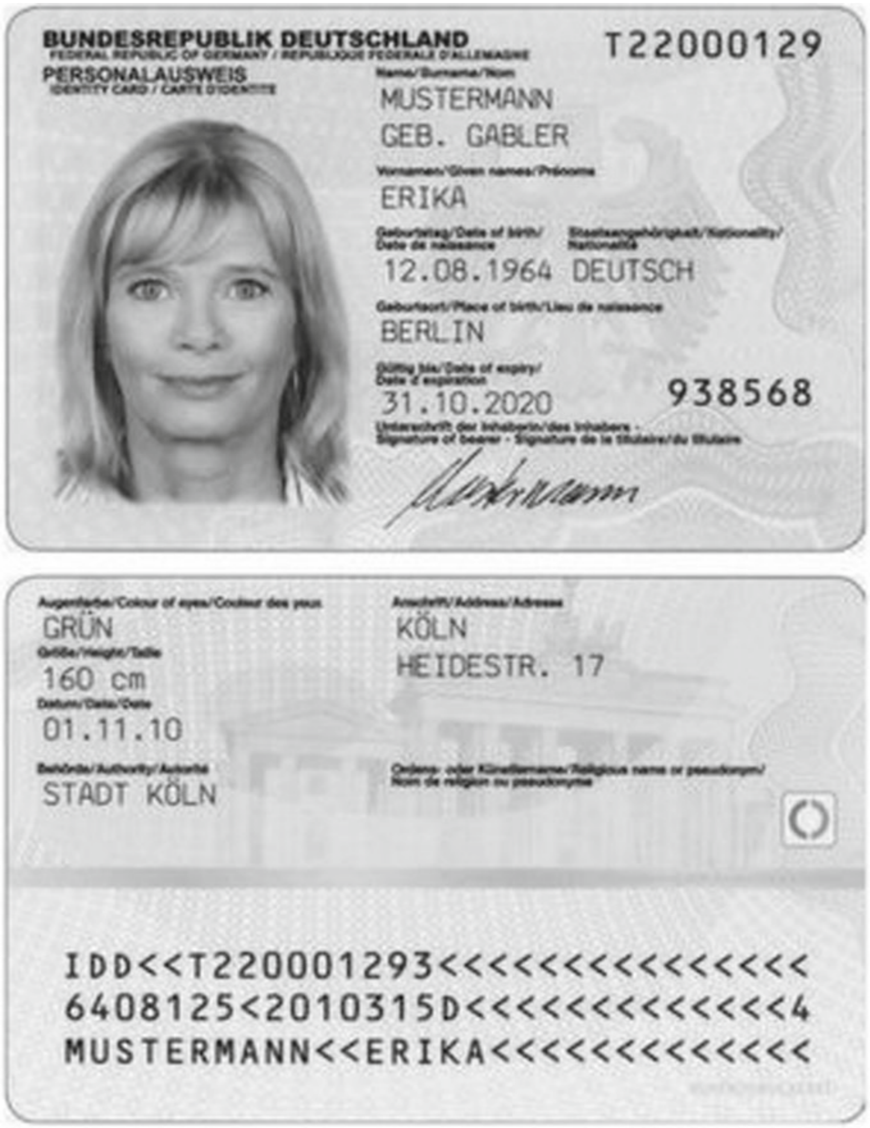
- Native Operating System;
- PACE (Password Authenticated Connection Establishment) protocol to encrypt the contactless communication and to avoid skimming and eavesdropping;
- Restricted Identification: Restricted Identification is a method to generate a pseudonym based on the individual nPA chip. A pseudonym can be used for the so-called terminal sector. The terminal sector is an identifier shared by all terminals of a certain service provider. This allows an (authenticated) terminal to recognize a chip based on the pseudonym previously received from the chip without reading out any personal data [1].

14.2.4.4 Macao Electronic ID Card (MEID)

Macao Special Administrative Region (SAR) is an administrative division of the People's Republic of China (PRC). Hong Kong is China's second SAR. Both SARs issue their own ID cards.

Features:

- since 2002;
- Java Card;
- digital signature application;
- "ID" application (card application containing personal data and authentication functions);
- main use case is for crossing the border between Macao and Mainland China or Hong Kong;



BUNDESREPUBLIK DEUTSCHLAND
FEDERAL REPUBLIC OF GERMANY / REPUBLIQUE FEDERALE D'ALLEMAGNE
T22000129
PERSONALAUSWEIS
IDENTITY CARD / CARTE D'IDENTITE

Name/Surname/Nom
MUSTERMANN
GEB. GABLER
Vornamen/Given names/Prénoms
ERIKA
Geburtstag/Date of birth/ Date de naissance
12.08.1964
Staatsangehörigkeit/Nationality/ Nationalité
DEUTSCH
Geburtsort/Place of birth/Lieu de naissance
BERLIN
Gültig bis/Date of expiry/ Date d'expiration
31.10.2020
938568
Unterschrift der Inhaberin/des Inhabers - Signature of bearer - Signature de la titulaire/du titulaire
[illegible]

Augenfarbe/Colour of eyes/Couleur des yeux
GRÜN
Größe/Height/Taille
160 cm
Datum/Date/Date
01.11.10
Behörde/Authority/Autorité
STADT KÖLN
Anschrift/Address/Adresse
KÖLN
HEIDESTR. 17
Ordens- oder Künstlername/Religious name or pseudonym/ Nom de religion ou pseudonyme

```
IDD<<T220001293<<<<<<<<<<<<<<<
6408125<2010315D<<<<<<<<<<<<<4
MUSTERMANN<<ERIKA<<<<<<<<<<<<<
```

Fig. 14.19 German e-ID

- e-Government functions;
- optical personalisation technology: laser engraving
- Multiapplication card for third parties;

14.3 E-Passports

14.3.1 Introduction

The traditional international travel document, the passport, allows identification of travellers by visual verification: a photograph printed into the passport is compared to the actual appearance of the person claiming to be the correct passport holder.

The effectiveness of such border control procedures depends essentially on the identification security provided by the ID document and the verification process. It is limited by two issues: the genuineness of the ID document itself and the data contained in it, and the reliability of the visual verification process performed by the officer.

Over time, sophisticated security features have been developed and implemented in travel documents to increase the technical difficulty for counterfeiters. This has reduced the risk of unauthorised persons creating counterfeit documents. The identification security itself is, however, still limited by the verification skills of a human officer and the imagination and energy of criminals to change their identity and/or adapt themselves to the appearance of other individuals to perform an impersonation attack.

Growing international terrorism, organised crime and increasing illegal migration have driven international efforts to implement processes, technologies and systems for improved secure identification. These efforts are based on essential progress in the fields of biometrics, microprocessor technology and cryptography. Biometric data of the passport holder, such as facial images and fingerprint data, are stored in machine readable form in a contactless microprocessor chip integrated into the passport document. Access to these data, along with assurance of their integrity and authenticity, is protected by advanced Public Key Infrastructures (PKIs). Thus, the travel document itself, as well as the verification techniques and processes, has been enhanced to allow secure identification of travellers. The traditional passport, allowing visual identification, has been upgraded to a platform for electronic identification based on unique biometric features of the passport holder. This upgraded document is the electronic passport (e-Passport) or electronic machine readable travel document (e-MRTD).

14.3.2 Constituents of Passports

A modern passport mainly consists of the booklet (cover and inner visa pages), the data page and the microprocessor chip with antenna.

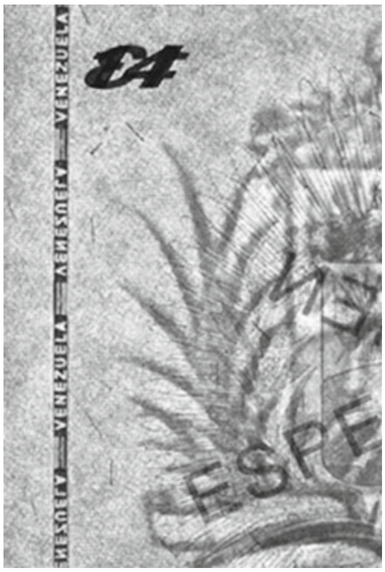


Fig. 14.20 Example for a mould-made watermark and country code security threads (*Source* Giesecke & Devrient/Veridos)

14.3.2.1 Passport Booklet

Due to the sophisticated production technique and integrated security features, paper used for passports differs from all commercially available paper. The substrate consists of a mixture of cotton and cellulose, without use of optical brighteners. A grammage of 90 g/m^2 for the inner pages is chosen in most cases.

The security paper used for passports contains a watermark incorporated using the cylinder mould technique during the production process. Mould-made watermarks present an insurmountable obstacle to counterfeiting methods relying on scanners and colour copiers. Unlike the two-dimensional watermarks produced on Fourdrinier machines, a mould-made watermark is characterised by finely modulated transitions that create a plastic, three-dimensional effect (see Fig. 14.20 for an example). Either in one precisely registered position or distributed over the full area of the paper, this distinctive feature quickly becomes familiar to users and is easily recognised. Mould-made watermarks have proved virtually indispensable for effective document authentication. Therefore, their use is recommended by Interpol.

Inner pages sometimes contain additional security features such as country-specific security threads, chemical sensitising or fibres:

Chemical sensitising:

Using chemicals such as alkalis, bleaching agents and organic solvents in an attempt to remove entries from documents activates an invisible colourant in the paper. The result is a discoloration of the paper that makes the alteration attempt obvious. Figure 14.21 shows an example of a chemical sensitizing of paper.

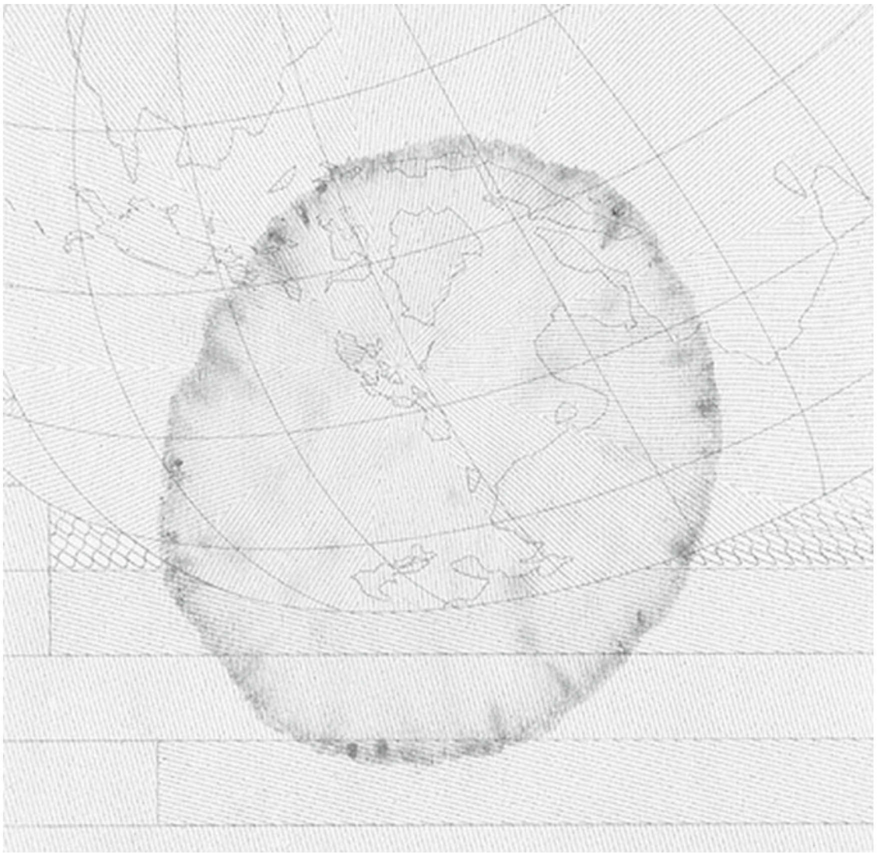


Fig. 14.21 Chemical sensitising of paper (*Source* Giesecke & Devrient/Veridos)

14.3.2.2 Passport Data Pages

Most passports still contain a paper-based data page. These pages are usually laminated with an overlay foil to protect the personalised information and to prevent forgery. Increasingly more and more countries are moving to plastic data pages made of polycarbonate which are supposed to be more stable and secure. The chip and the antenna could be located in such a plastic data page which offers both maximum protection. The foremost target for forgery or alteration attacks on travel documents is the data page. This is why the printing methods used for the blank data page and for personalisation are configured to form an integrated security design comprising features which are effective at all authentication levels. Examples for security features on paper-based data pages are intaglio printing, optical variable inks or protection foils with Diffractive Optically Variable Image Devices (DOVIDs). Plastic data pages can be equipped with security features similar to those available for polycarbonate ID cards.

14.3.2.3 Microprocessor Chips for Passports

According to ICAO, 32 Kbytes memory is the minimum storage requirement for e-Passports. This is enough to store the MRZ, the facial image (approx. 20 Kbytes) and security elements such as the document security object (EF.SOD) that contains the hash values of all data groups used. The precise size depends on the country-specific requirements. There is no limitation for memory capacity defined. For some non-EU countries, it is enough to store the minimum ICAO requirements for e-Passports and the US requirements for VISA Waiver countries. Please note that a chip is recommended by ICAO but is not mandatory for passports. The so-called LDS (logical data structure) is defined by ICAO. The currently[1] recommended version is LDS 1.7. It contains a number of mandatory and optional Data Groups (DG):

DG1: Contains the details recorded in the MRZ (mandatory)
DG2: Encoded face (mandatory)
DG3: Encoded fingers (optional)
DG3: Encoded eyes (optional)
DG4–7: Displayed identification feature(s) (optional)
…
DG15: Active authentication public key info (optional)

DG4 to DG16 are optional, and the issuing state or country can decide about the usage of these data groups [2].

14.3.3 EU and ICAO Requirements

ICAO Specification DOC 9303 part 4 defines passport requirements to ensure international interoperability and in particular sets standards for security.

By Article 6 of EU Council Regulation (EC) 2252/2004 of December 2004 (and additionally adopted technical specifications), standards for travel documents are defined: the member states decided about security features and biometrics in e-Passports.

The European Commission set the implementation time frame so that all member states had to implement the facial image requirements by 28 August 2006 (as a consequence, all member states also fulfilled the US requirements for the VISA Waiver countries to issue biometric enabled passports by October 2006). Even though not all EU member states met the deadline, the vast majority did so within 2007.

The protection of the facial image is ensured by "Basic Access Control". This requires the reading of the MRZ in the passport to gain access to the data on the chip.

[1]May 2016.

As a next step in the EU, the additional storage of two fingerprints on the passport chip was required from 28 June 2009. The EU Commission considered these data as more sensitive and decided to protect the fingerprint data using "Extended Access Control", a system which works with a PKI (public key infrastructure). Currently, only EU member states are supposed to have access to the fingerprint data. But at publication time of this book there was no EU wide system for exchange of certificates to access fingerprints of passports of EU citizens. Whether access to other countries will be granted has to be decided by the EU at a later stage.

The third generation of European e-Passports has just been introduced. From the beginning of 2015, EU passports have to include Supplementary Access Control (SAC) which is also recommended by ICAO. The used protocol is PACE which is replacing BAC in the long term. For global interoperability, states must not implement PACE without implementing Basic Access Control until 31 December 2017. Inspection systems should implement and use PACE if provided by the eMRTD chip [3].

14.3.4 Security Protocols

E-Passports are equipped with a contactless RF chip containing the digitised biometrics of the holder. The authenticity, integrity and privacy of the data stored in the e-Passport chip must be strongly protected. Therefore, ICAO has specified some security mechanisms. The most important ones are listed here:

- PA: Passive Authentication (digital signature to provide authenticity);
- AA: Active Authentication (challenge response to provide integrity);
- BAC: Basic Access Control (authentication and secure channel to provide privacy); and
- PACE: Password Authenticated Connection Establishment) (authentication and secure channel to provide privacy).

Passive Authentication (PA) provides proof of the authenticity of the e-Passport by verifying a Document Signer Certificate and the electronic signature on the stored data. Currently, Passive Authentication is the only security mechanism mandated by ICAO. In an e-Passport equipped with this feature, the data can be validated to verify its authenticity but chip cloning (i.e. copying the complete data stored on one MRTD chip to another chip), or eavesdropping cannot be prevented. PA is the only mandatory security protocol in e-Passports [3].

Basic Access Control (BAC) should ensure that data can be accessed *only* with the consent of the e-Passport holder to prevent skimming and eavesdropping. Although BAC is only optional according to ICAO, there is worldwide no e-Passport being issued without BAC. Access to the data is enabled after successfully reading of the printed Machine Readable Zone (MRZ) with an inspection system to generate a document-specific access key. This key is derived from parts of the MRZ (date of birth, expiry date and serial number). Using this access key, the inspection system authenticates the e-Passport and a session key for the protection of the communication channel is generated. Based on symmetric cryptography and due to a limited strength (entropy) of the keys used, BAC is a dated technique [4].

PACE is a password-authenticated Diffie–Hellman key agreement protocol that provides secured communication and password-based authentication of the eMRTD chip and the inspection system [4]. One special feature is that a strong session key is generated independent of the strength of the password. Therefore, the MRZ is enough to generate strong encryption. In case of ID-1 cards, the usage of a Card Access Number (CAN) is recommended. The CAN (6 digits are recommended) can be printed on the front page and can be used instead of the MRZ.

Extended Access Control (EAC) restricts access to particularly sensitive data groups in the LDS, such as fingerprint data, to authorised parties. Accessing and using such data is only possible when the potential user, such as an inspection system at a border control station, has a valid digital certificate and associated private key at its disposal. Such certificate(s) are generated and administrated by the country issuing the e-Passports.

Moreover, using EAC provides secured data communication by the secure authentication of the inspection system and by the means of strong (asymmetric) encryption. Implicitly, the EAC protocol also prevents "chip cloning". EAC is mandatory in EU member states and has been introduced in a large number of additional countries.

Active Authentication (AA) ensures the uniqueness and authenticity of the microprocessor chip within the e-Passport. This effectively prevents "chip cloning".

14.3.5 The ICAO PKD (Public Key Directory)

Based on the Passive Authentication (PA) mechanism, ICAO installed a worldwide eMRTD Public Key Infrastructure (PKI). This directory enables the verification of digital signatures on electronic passports. By verification of the signatures and the signed hashes in the Document Security Object, the authority verifying the document can be sure that that signed data are authentic and have not been modified.

Certainly, the PKD can also be used for revocation of certificates in case an e-Passport is reported stolen. In May 2015, there were 46 participating countries. Regarding the number of more than 105 countries issuing electronic passports, this is a small number. The verification of the Document Security Object is an essential part of the verification of genuineness of a passport.

14.3.6 LDS 2.0—The Future Passport

The current e-Passport enables automated border control and enhanced the general passport security. But there is one issue which has not yet solved by the introduction of smart travel documents, the secure and automated verification of visas. Visas are still issued as a stamp or as a sticker. They do not contain any RFIDs and therefore can only be read and verified by optical scanners or the border staff. Automated gates nowadays cannot be used by third-country nationals holding a visa of the country they are visiting. Holders of Schengen Visa or Schengen residence permits cannot use e-Gates when entering Europe.

Things will change when LDS2.0 replaces the current e-Passport and when it is in use in large scale. The LDS2.0 passport will additionally store time stamps and electronic visas. At publication of this book the specification was still under construction at ICAO, but first interoperability trials have been executed in 2016.

14.4 Conclusion

Security mechanisms and features described above constitute the state of the art of ID document security. Given the growing importance of secure identification in daily life, in the real world as well as on the Internet, new technologies will be developed and incorporated into ID documents in order to stay ahead of the counterfeiters and to make identifying an individual more easy and secure.

Making ID cards and passports smart also enabled automated verification, which leads to more efficient and faster processes. At borders, ABC Gates (Automated Border Control) leads to shorter waiting times for passengers.

References

1. BSI Technical Guidance document, BSI-TR-03110-Part-2-V2-2.
2. ICAO Doc 9303 Part 10: Logical Data Structure (LDS) for Storage of Biometrics and Other Data in the Contactless Integrated Circuit (IC).
3. ICAO DOC 9303 Part 11: Security Mechanisms for MRTDs.
4. Introducing the PACE solution by Dr. Jens Bender & Dr. Dennis Kügler (BSI) in Keesing Journal, Issue 30, 2009.
5. ICAO Doc 9303 Part 5: Specifications for TD1 Size Machine Readable Official Travel Documents (MROTDs).

Chapter 15
Smart Card Technology Trends

A Review of the Variety of Applications and the Market Drivers

Chris Shire

Abstract This chapter examines the historical use of technology in smart cards and the trends in the future. It considers the options that are available, the choices that must be made with a smart card scheme, the issues that affect the design of the card and its applications. The influence of consumer demand is discussed and the drivers that will define the use smart card technology in the future.

Keywords Silicon · Technology · Trends · Memory · Microcontroller

15.1 Trends in Smart Card Technology—Today and the Future

Today, the most common token of a person's digital rights is a smart card. Smart cards with a variety of silicon security chip technologies are used across a whole range of applications from personal use, including bank cards and mobile phones, to enterprise systems such as computer systems, and building security. They provide the credentials to diverse security processes which are likely to be even more sophisticated in the future with the increase demand for remote authentication of the user and their terminals via the Internet. Inside many laptops and notebooks, secure silicon based on chip card Integrated Circuit (IC) technology, in the form of a Trusted Platform Module (TPM) or Subscriber Identity Module (SIM), is already authenticating the access to both wired and wireless networks. In many open networks, every device (client) as well as the server has to be secured against different security profiles, so there is not a one-size-fits-all of security. In many applications, especially those using personal data, the focus is now on end-to-end security involving a variety of secure devices. However, deployed products need to be able to evolve to meet the constantly changing threats from malware.

Consumer appliances from TV set-top boxes to games consoles use secure "smart card" ICs to protect content usage. Integrated circuits for smart cards generically

C. Shire (✉)
Infineon Technologies UK - AIM CC, Munich, Germany
e-mail: chris.shire@infineon.com

K. Mayes and K. Markantonakis (eds.), *Smart Cards, Tokens, Security and Applications*, DOI 10.1007/978-3-319-50500-8_15

called “chip card ICs” can be assembled in various formats to become the root of trust for physical or logical access, content control, or as an embedded solution to ensure device to device authentication in a single system, or in a distributed Internet of Things (IoT) network. To understand how the smart card of the future could develop, the history of smart card development must be considered, and how it has been influenced by technology, and the services it supports. In some cases, the services have driven the demand for more advanced technology; in others, only the availability of technology has enabled the service to be rolled out. On the other hand, there are underused technologies in place today, some which have yet to fulfil the needs of the service providers, e.g. convenient secure biometrics, and copyright protection, where security technology has yet to be accepted by consumers.

15.1.1 History

Chip card ICs concepts were invented and patented in the early 1970s. The earliest services were for replacing cash in remote payment applications such as pay phones and vending machines; this reduced the cost of cash handling and theft from unattended and sometimes isolated locations. The first mass use of the cards was for French public pay phones. The French government was, and still is, an enthusiastic supporter of the technology. French companies have led the way in developing designs from the original simple memory cards to the more complex microcontroller cards with read/writeable memory and data processing capabilities. With the advent of the PC and data networks, a secure token in the form of a smart card for authentication became a useful tool in diverse applications from health insurance cards in Germany to loyalty cards across Europe.

The advent of Pay-TV in the early 1990s demanded more sophisticated designs, as did fraud-resistant banking cards in France and were major drivers for this infant industry and led to the continuing leadership of that country’s industry in the global market. Early innovations included electronic purse systems, in which monetary value was stored on the chip, so that point of sale terminals accepting the card did not need network connectivity. These were tried throughout Europe from the mid-1990s most notably in Germany (Geldkarte™), Belgium (Proton™), the Netherlands (Chipknip™ and Chipper™), Finland (Avant™) and Denmark (Danmønt™). But the advent of low-cost online payment processes and high cost of dedicated e-purse infrastructure and services charges limited the take-up of them in most countries.

Chip card IC design has really been has been driven by demand for Subscriber Information Modules (SIMs) for GSM mobile phones since their inception in the late 1990s. Cloning of phones had been a problem with analogue mobile phones in the 1980s, and various proprietary security protocols and ICs were developed, but were handset and network-specific. The standardisation by European Telecommunications Systems Institute (ETSI) [14] of a security element in the SIM was a huge step forward. The take-up of GSM was relatively slow to begin with. Sales were focused on phones on contract to business users, so restricting the market growth.

The introduction of prepaid services controlled with the SIM led to an explosion in consumer demand, far beyond the early estimates. In the UK, in the past fifteen years, the number of subscribers has grown from under 6 million to over 80 million [12]. Many people have more than one handset, and the use of data telemetry systems has led to subscriber penetration greater than the human population. The demand for SIMs will continue for the foreseeable future, even with saturation of the established markets. The GMSA has predicted that the current worldwide estimated 3.6 billion unique mobile subscribers at the end of 2014 will grow to by another billion by 2020 [22]. The usage of SIMs is even higher due to a relatively high rate of churn. Many users frequently change their networks and phone contracts which increases demand for SIMs, which was already approaching 5 billion cards in 2015. One major reason for this is the business model in developing countries such as China, one of the largest markets, where many people buy several prepaid SIM cards per year rather than using an account-based system. The number of fixed long-term contracts in China is increasing, but it is a largely cash-based economy, so this demand pattern will only change slowly.

The historical business of prepaid public phone card chips has slowly faded in the last decade as mobile technology has become the communications system of choice, even more so in the developing world. The growing deployment of the mobile phone with SIMs has increased demand for more sophisticated features. The earliest SIM designs needed no more than 8 kilobytes of Electrically Erasable Programmable Read-Only Memory (EEPROM), but this has increased many times over with more sophisticated software such as SIM Toolkit. This platform supports control of services and features like Over-The-Air downloads, and content such as music, video updates, e-mail and Internet access; the memory demand has grown. Given the rapid return of investment on mobile services, this has led to the need of the largest memory on a SIM available being used as soon as possible. Historically, the SIM memory size doubled every two years, but as Wi-Fi usage increases and data services such as 3G and 4G services become prevalent, this trend may level out.

In other smart card market sectors, there are different trends as the service deliverables have other use cases. Volume demand in the finance sector is increasing as card fraud is a present and growing threat. But payment systems although dominated by the Europay, Mastercard, Visa (EMV) specifications [4] that have been developed since the early 1990s, there is a diverse range of implementations for credit and debit cards, with several national and regional variations. The EMV specification defines the interaction at the physical, electrical, data and application levels between chip card IC cards and card processing devices for financial transactions. Large portions of the specification are based on the chip card IC interface defined in ISO/IEC 7810 and 7816. The first version of the system was released in 1994, and several upgrades have followed as the services and technology required. But since the banking services market is essentially an online business, the security element required for EMV transactions requires limited memory and security features. The typical EMV card has perhaps 8 kilobytes of user Non-Volatile Memory (NVM) with symmetric and asymmetric cryptographic functions. It is only when additional services or multi-application cards have been required has the chip size and functionality expanded.

The result is that EMV banking cards have seen a price decrease year on year since their introduction, such that chipped cards now cost little more than their un-chipped predecessors of 10 years ago.

Smart cards with contactless interfaces are becoming increasingly popular for low-value payment and ticketing applications such as mass transit. Visa and MasterCard have used the ISO 14443 contactless protocol in conjunction with EMV cryptography. This is currently being rolled out across the world. Across the globe, contactless fare collection systems are being implemented to drive efficiencies in public transit. Japan, perhaps the world's most technophilic society, has taken to using contactless products in many aspects of everyday life. In London, over 30 million public transport "Oyster™ cards" have been issued, enabling billions of journeys every year. This trend is likely to increase as more UK transport networks connecting to London start to use the technology, although contactless EMV cards will replace many proprietary schemes cards. Across the world, various contactless ticketing specifications that have emerged are often local in use and function, for example the various "Calypso™" transport ticketing schemes issued in several cities in France but were not interoperable between the different city schemes. This will change with time as new regional standards are developed, so one day in the future it could be possible to load tickets onto a Near-Field Communication (NFC) phone in the suburbs of Birmingham and travel to a suburb of Paris through automatic gate systems, but it is unlikely that there is an economic justification to extend such through ticketing to every smart card or to every city.

The diversity of services leads to a wide variety of designs and range of capabilities; the result is a plethora of chip designs with memory demands from 200 bits to over 2 megabytes of NVM. However, the vast majority of cards issued today are using under 100 kilobytes for data with simple data processing and limited encryption. Another driver is the storage of credentials which enables the identity of the card holder to be authenticated. This leads to two-factor authentication processes where the presence user can be verified by the credentials in conjunction with a passcode. Third-factor authentication with the use of biometrics can provide even more secure transactions, subject to issues of human factors of convenience and system reliability. There is a wide demand for identity management in many closed user groups, from libraries to health insurance cards for ease of access. Identity theft and remote data access are driving the need for secure nationally used identity schemes, such as driving licences, ID cards and electronic passport solutions. In the future, even more secure silicon will be supplied in non-card formats to support user and device authentication for online systems, such as games, content rights and telemetry. The embedded smart card IC market may end up even larger in volume than the SIM business. One example is authentication of things. Although not traditionally part of the smart card market, because of counterfeits, there is a huge rise in securely identifying objects connected to cloud-based servers, as part of an IoT infrastructure. These simple tags are often not very secure and similar in function to that of a basic memory smart card, but provide enough security to protect the originality of many consumer products.

15.1.2 Technology Choices

The choice of chip in card is sometimes ignored; there have even been cases where the colour or shape of the plastic card has been seen as a more important card feature. But there are characteristics of the chip which matter. In some cases, chip features are promoted to consumers, such as the size of memory or performance. However, it is more important to the issuers as to how a particular chip hardware and software perform together. Several key questions need to be considered, such as the security protection profile of the chip, the performance of the device, and the chip compatibility at the physical interface level.

In the past, "security by obscurity" was common, and each design was in theory bespoke, but this in turn limited migration of design, so compromises were made. Finding such a compromise on a new chip, whose design was based on a previously "cracked" chip, made hacking almost a sport. The risk of attack versus the cost of the solution needs to be carefully considered and impacts the choice of technology. In the case of many commercial applications, simple security techniques such as limiting system access to only those users holding a card with a registered unique ID number is considered sufficient. Since most chip card ICs come with this feature as standard, the security is not a defence against spoofing by a card emulator, but with the addition of a PIN, held off card, seen as acceptable. Further tactics such as traffic analysis on the usage of the card, i.e. a card cannot be in two places at the same time, or the use of a time stamp in the memory can reduce spoofing in a networked environment. But for offline use, cryptographic techniques maybe needed to ensure data integrity and for more high-risk environments a formal security analysis maybe required. A professional risk assessment and security audit according to BS7799 and ISO/IEC 17799 should identify the issues, from which the type and features of the security required on the card can be developed. For certain government applications, more formal security certification is mandated. The security of chip can be independently assessed according to Common Criteria [15] which has been used for all types of information security. For chip card ICs, the most recent protection profile is PP0084 [2]. Devices have been developed which have been successfully evaluated to the level of EAL 6+ based on this profile. These devices include numerous security features to reduce vulnerability from attack, including non-standard CPU designs, sensors to protect the silicon from various energies directed at the device and using hardware and software encryption throughout the design. Software to run on these chip can be security-certified up to EAL6. The EU directive on digital signatures [6] has led to Common Criteria certification to EAL4 or above being demanded for the chip, the operating system, and the application in many government- supported schemes. The cost, in terms of time and money, can impact a single project, but benefit can be gained if the protection profile used allows for reusability of existing elements for other developments. In the USA, the government has mandated that their own Federal Information Processing Standards (FIPS 140) [16] is used for cards it issues, and this standard has been taken up by private enterprises.

Performance criteria are sometimes assumed, but often system-level issues will be affected by the chip, e.g. ensuring the software protocol is optimised for the readers it will be used with. This has come to light with transport operators of train stations and buses requiring gate interaction time requirement of 300 ms or less. If a passenger is held back any longer, they maybe believe the system has a fault, and may try again, this can cause disruption in flows of people at gates and even impact the reputation of the operator. The physical limitations are in some cases are card scheme specific, to maintain some kind of interoperability of the electrical and physical formats of the card as specified by ISO/IEC 7810 and ISO/IEC 7816 to ensure ease of use. In the past, the low level of interoperability limited usage in many instances; commands for one card could not be understood by another as the Application Programming Interface (API) was specific to the chip firmware or the operating system of the card designer. This has become less of an issue as APIs have become standard, such as the Java Card specification [18].

The choice of chips for smart cards has been limited by various factors. The cost of the card has been seen as a key factor, often because it is treated as an operational expenditure by the provider of the service. But the cost of cards maybe reduced into insignificance by the costs of the deployment, infrastructure and managed services. For example, over 5 million employee cards have been deployed by the US Department of Defense, the so-called Common Access Card (CAC) [3]; they are believed to have cost under 10 USD each, i.e. a total of around $50 million. The system costs, however, are estimated to have cost over $1 billion so far. The CAC enables encrypting and cryptographic signing e-mail, facilitating the use of authentication tools, and establishes an authoritative process for the use of identity credentials. In some cases, further costs may be incurred if a card scheme operator wants to offer a range of different cards and functions over a period of time. This requires sophisticated card management. From an information technology network system perspective, cards are essentially offline remote thin client nodes. As such, these nodes are only in the network infrequently, and often randomly. However, the users require instant service when online and may not be conducive to any system updates which change their interaction with the service. If a change to a card is proposed, or even the card is to be replaced and upgraded, the return on investment can be difficult to calculate. It is not just a simple cost versus revenue calculation, but may involve improved efficiency in data bandwidth management, reduced transaction times and faster backend processing. Another example of card scheme impact on indirect costs is the more than 180 million Chip and PIN [7] EMV banking cards in circulation in the UK; the cost per card including issuance can be less than £1 each. The cost of setting up the national system of over 1 million readers and infrastructure has been reported to be over £1 billion. However, the current fraud reduction from lost, stolen and counterfeit credit cards saved per year is over £200 million each year compared with the same figures from 10 years ago [21].

If the choice of chip card IC can be determined, another issue is the lifetime of the product. There have been instances when a card has been deployed for a project, only to find soon afterwards the chip card IC was at the end of its production and no further directly compatible cards could be produced. Some so-called existing "standard" cards will not operate with newly introduced "standard" readers; this is especially true of contactless products where operational and testing requirements are still evolving. If a bespoke solution is chosen, the issuer's problem is to manage the long-term support and migration for the software, and any other intellectual property used in the design. Even with testing against all known environments, a change in the host middleware, quite remote from the card, may leave the system inoperable if software drivers are not maintained. To mitigate this risk, one option is to ensure the supply of cards and chip, and supporting technology can be supported unchanged for the duration of the lifetime of the whole system. An alternative is to choose a design which can be upgraded in future, by using an open system specification. The first option can be expensive, not just for the upfront investment in product and storage, but to manage the infrastructure around the card. There may be an issue of software driver compatibility in the host. Operating systems in networks are updated regularly; the software driver needs at the reader interface to present the same API, both logically and dynamically. Forecasting long-term requirements is often an educated guess, and there several instances of simple "upgrades" which have been very costly to recover from. The option of working to a defined open specification can be costly in the short term and time-consuming if several parties are involved. But the benefits from long-term open interoperability have proved to be enormous.

The most successful example of an open interoperability is the ETSI GSM SIM card specification "GSM11.11" for testing of compliance of cards and mobile handsets, so that the operating system and the underlying chip hardware have be rendered academic, at least at the logical level. The specification has required several updates over the years. In some cases, it has been seen as a hindrance to new developments, but with over several billion compatible SIMs issued every year it is unbeaten. Across the world, there are few other smart card schemes which are truly interoperable. One is the EMV Payment Card specifications. EMV financial transactions are more secure against fraud than traditional credit card which use the data encoded in magnetic on the back of the card. This is due to the use of encryption algorithms to provide authentication of the card to the processing terminal and the transaction processing centre. However, processing can be slower than an equivalent magnetic stripe transaction, due to the end to end cryptography overhead. Although not the only possible method, the majority of implementations of EMV cards and terminals confirm the identity of the cardholder by requiring the entry of a PIN (Personal Identification Number). In the future, systems may be upgraded to use other authentication systems, such as biometrics. Some schemes using fingerprint verification of pre-authorised debit or credit card account details via an online terminal. One of the most current schemes is "Apple Pay™" linked to the fingerprint sensor on the phone. This is slowly becoming an accepted alternative to carrying a card in an online retail environment for some people. However, there are many obstacles to taking a proprietary biometrics scheme and scaling it up to a global system.

The need for electronic identity cards and tokens has led to a variety of schemes. One exception with an open interface is the US government General Service Card (GSC) specification. Generated by its standards bureau NIST the NISTIR 6887 specification has sought to generate and mandate for all major federal programs the smart card interface both physical and electrical, a so-called "card edge specification". It has been developed further as the Personal Identity Verification (PIV) specification [1]. It has also lead to the ISO/IEC 24727 smart interoperability framework standard developed by ISO/IEC JTC1/SC17 WG4/TF9 working group [11]. In this way, future generations of cards should be compatible with the existing terminal infrastructure and vice versa. Such specifications have the added benefit that multiple suppliers can design compatible products, and given some sort of compliance testing resulting in economies of scale and improved commercial terms for the card scheme operators.

In Europe, there exist several such "open" specifications in different sectors, one of the most active being the transport industry. The "Digital Tachograph" project, which took over 10 years to develop, is deployed across the EU [9]. A key gain was the testing for compatibility of the readers and cards. This was ensured by designating a single test house in Italy to ensure there could be no question of misinterpretation of the requirements. However, the need for interoperability depends on a sound underlying business case. For example, despite the proposed EU wide transport ticketing standard for applications InterOperable Public Transport Application (IOPTA) [23], this has never been implemented between countries. Individual countries have adopted major portions of the standard, such as in the UK with the ITSO specification [19]. Here again testing for compliance has been a key requirement of the design, and ensuring that the security of transactions can be managed across distributed and remote networks.

At the global level, the only identity scheme using chip card IC technology and biometrics is the ICAO e-Passport. The International Civil Aviation Organization (ICAO) generated a specification [17] for electronic passports which may eventually apply to all 188 countries of the United Nations. The specification is based upon a series of ISO standards such as ISO/IEC 7501 for the format, ISO/IEC 14443 for the contactless communication and ISO/IEC 10373 for testing protocols. A series of interoperability test sessions, financed by various governments, were set up. The requirement derives from mandates of the US Government with its Patriot Act [20] and the EU directives [5] for visa-free travel. They are applied to travellers from their own countries or foreigners that are from countries which are acceptable politically and have a suitable passport vetting process. Such travellers only need a biometric e-passport based on the ICAO specification. Travellers from non-compliant and unacceptable countries will still require a visa. Following the US mandate, over 30 "visa waiver" countries were ready with their first e-passports ready by October 2006. To ensure compliance, a combination of government-funded and several independent test houses have been set up. The governing model here is different from previous examples; although ICAO can make recommendations, it has no power to enforce compliance. The only influence is a combination of peer pressure and rule of law in some countries to have a compatible system. The effect of these mandates has been both huge political and commercial pressure to ensure the technology was ready to meet the deadline.

The ICAO specification has three major influences on smart card technology; it has helped to standardise contactless technology; in chips, software designs, and readers to become completely interoperable. The chip was specified to have at least

32 kilobytes NVM and typically around 80 kilobytes. Many countries chose larger memory sizes to allow for fingerprint biometric images to be included. In the past, many contactless systems have been closed local systems. For the first time, the ability to take the same smart token from place to place across the world and the data to be read and verifiable has had a huge impact on systems integration and identity management. There are now over 100 countries introducing such passports, whose citizens represent 90% of the world's travellers. It has been proposed that in future that the chip in an e-Passport could be used to contain other information such as travel visas or entry stamps as an alternative to the online-stored credentials used online by e-Visa schemes and e-Border databases for entry records. The US has been taking ten fingerprints from every traveller arriving by air into the country since 2008; in future, such data may be stored on the chip. This would require larger memory in excess of 200 kilobytes to store all the data. The technology to hold and process these data in an electronic token is available now, but implementation of such products is not expected for some years.

Passport systems have, in time, become the basis for a global public key infrastructure, allowing books and systems to be authenticated by different countries. Once a government- issued certificate becomes verifiable by third parties, the opportunity it presents as a root of trust for other activities such as ecommerce is enormous.

The features of e-Passports will lead to the basis for many systems. It will influence the design and demand for contactless reader and card technology for the foreseeable future. As an open standard identification design, it will influence other national identity cards schemes and even may replace proprietary designs used for local and even commercial identity management and access control systems.

15.1.3 Technology Drivers

The technology used of smart cards is determined by the physical interfaces to the chip and the type pf technology used therein. There are two basic chip types used in smart cards used to keep data secure:

- Memory: It can be compared to a storage disc in a personal computer, but of course lower capacity due to smaller lower complexity chips. Secure memory cards use either Non-Volatile Memory (NVM) such as EEPROM or Flash technology to store data. The data are secured by the use of a simple access code which controls access and in some cases by a cipher stream engine which verifies against a secured key. There are of course high-density Flash memory products in a variety of form factors such as Secure Digital (SD), MultiMediaCard (MMC) and others, but data are not held securely and so these are not considered smart card products.
- Microcontroller: It might be compared to a tiny computer with storage, processing unit and temporary memory. Although such a chip needs extra silicon compared with a basic memory chip, the flexibility of such a design offers higher performance which can outweigh any cost differences. Furthermore with today's dense

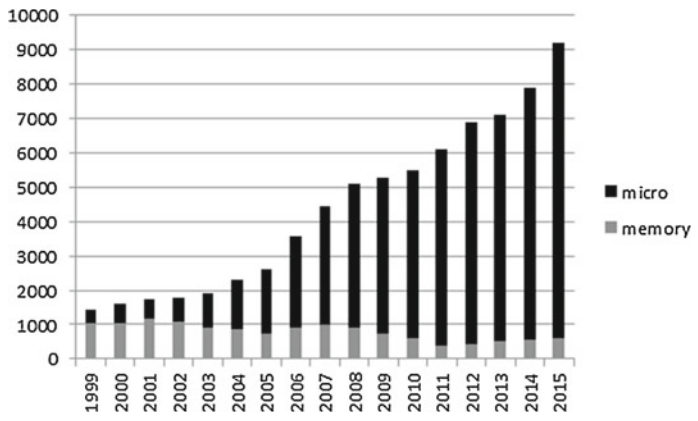


Fig. 15.1 Demand over time for smart card IC types—millions of pieces [27]

designs, smaller chip geometries and low gate count dedicated smart card controllers; the overhead can be measured in terms of a very few square millimetres. Microcontroller cards typically have some memory (ROM or Flash NVM) for long-term program storage, e.g. the operating system, RAM for temporary data processing and NVM for extra program storage, e.g. applications, card-specific elements such as card user details and issuer credentials. In some designs, chip may have only RAM and Flash. This type of design allows the operating system itself to be programmed at the point of issuance or even updated in the field, for example Over-The-Air in a mobile phone. At present, smart card designs combining two chips, and very large mass storage Flash memory chip and a secure microcontroller, have had some limited use. There have been security issues in the past over the use of Flash technology, because chip designs based around standard memory design have been successfully attacked using probing and eavesdropping methods. With modern chip design, security features have been incorporated into the Flash to defeat such attacks [24].

Chip types, memory, and microcontroller use technology which has been optimised for smart card market use, with low power consumption and a variety of dedicated features to add security. These two fundamental varieties are unlikely to change in the future. The earliest memory cards capacities were a few hundreds of bits. The largest memory cards at the time of writing have in excess of 256,000 bits. Historically up till 2003, the volume of memory chips sold worldwide was higher than microcontrollers, as shown in Fig. 15.1. But by 2015, of the 9 billion plus chips sold, only around 7% are of just a few memory types, whereas for microcontrollers there are dozens of types, but they are based on a handful of logical architectures. Since then, there has been a large decline in contact-based memory cards, but contactless memory cards have grown considerably for transport and access control.

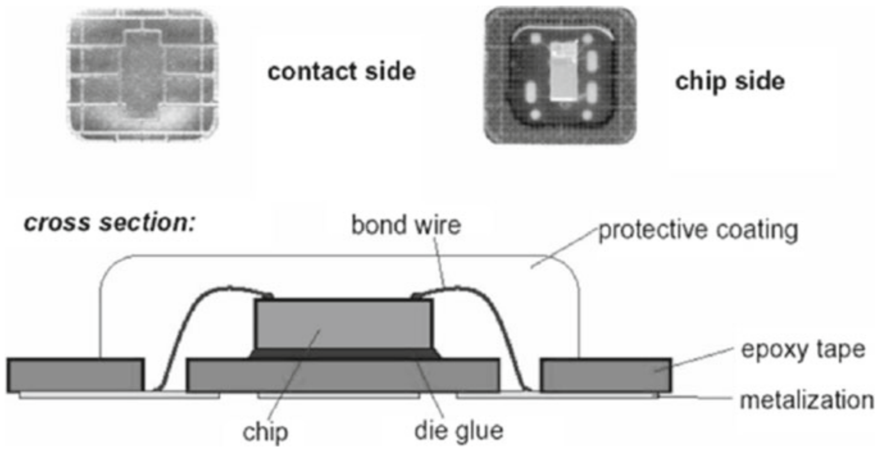


Fig. 15.2 Smart card module construction—courtesy of Infineon Technologies AG

There are two types of smart card interface; contact based, taking in over 90% of today's market using a serial protocol similar to RS232 on a computer and contactless, using wireless, most commonly with a Radio Frequency (RF) carrier of 13.56 MHz. The price of a chip depends on the complexity of the chip performance, security and features, and therefore size of chip. The underlying silicon technology has changed considerably over time. But the original ISO standard has dictated the physical dimensions and as such the processing capability. The module size defined by ISO/IEC 7810 allowed for devices not much greater than 25 mm^2; this was to allow space for the bonding wires in the module as shown in Fig. 15.2, and because larger chips might crack during normal use. In addition, power consumption has been required to be less than 10 mA at 5 V for many applications. Based on silicon geometries greater than 1 μm, this inevitably limited the early designs to small memory and basic CPU functions. The speed of the serial interface limited the data bandwidth. Typical values were 19,200 bits/s, and this remains common today for many cards.

15.1.3.1 Memory Card

A typical card might have less than 1 K bytes of memory space, and unique chip number as part of the Answer To Reset (ATR) and with PIN-based protection for a non-volatile memory to store relatively static set of data. In the early contact-based memory cards, the most common function was a simple one-way counter which decremented a stored value, when instructed by a host terminal. This card was used to implement electronic money counter and used in a wide variety of systems where cash handling can be difficult, such as pay phones. At one time, over 100 countries issued such cards, as they offered a cheap e-purse type function which saved on cash management costs and reduced fraud. Several million a year were issued by British Telecom alone in the late 1990s until their popularity diminished with the uptake of mobile phones and payment by the use of banking card. Phone cards in fact have become collector's items, such as stamps, in some cases with extraordinary prices. Another early use case was for loyalty cards; the contact chip was a low-cost authenticator method of a user's privilege. However, one of the most successful loyalty card schemes in the UK, with around 15 million issued, was based on a memory card but has now moved to a barcode-based card with online authentication to reduce cost.

Contact cards are rarely used in mass-transit applications because of the wear and tear on the readers, although contact cards and readers are cheaper, maintenance costs can be prohibitive. Contactless memory cards have become the technology of choice for automatic fare collection card systems across the world, with many 10's of millions issued every year. Authentication is often offline with either passcode or symmetric key schemes linked to a Secure Access Module (SAM) in the terminal. There are, however, many contactless memory cards used by small closed user groups across the world for access to clubs, work and school canteens, vending machines and other token-based systems.

15.1.3.2 Microcontroller Card

With the introduction of small 8-bit microcontrollers in the late 1970s, it was inevitable that they would find their way into smart cards. Contact-based microcontrollers were originally used in applications where a memory card may have been used. But as extra memory capacity has been required, plus as services became more complex with enhanced security and on-card processing for application selection or data processing, microcontroller- based cards have dominated. There are three dominant CPU designs in smart cards, one based around the Motorola designs of the 6805, one based on the Intel 8051 core, though later enhanced versions are based on the Intel 80251, and in the past few years ARM M7 derivatives for high-end mobile security. Bespoke Reduced Instruction Set Computing (RISC) designs did become popular in the last decade, but due to the need by operating system developers to reduce development time, such bespoke designs have tended to fade away with the exception of some specialist applications like Pay-TV. In all cases, the designs have been heavily modified to ensure the data are held securely. This is a crucial requirement for a smart card. If the core used in a chip card was a standard microcontroller design chip, reversed engineering emulations were relatively easily developed. In the more secure designs, the architecture and circuit layout rules have been altered to ensure data cannot be extracted or deciphered easily. Devices originally developed for simple control applications are of course cheaper to modify for chip card IC use and so offer a fast return on investment. But card issuers found themselves compromised and left unable to protect revenues. In addition, standard designs, although cheaper, may have issues with transaction speeds.

Only secure designs with features like cryptographic co-processors to accelerate encryption algorithms can offer the performance needed. Security sensors are built into the chips to thwart attacks. The physical design is designed so that chips cannot be probed or eavesdropped electronically. Today, contact-only-based microcontroller cards make up around 75% of the annual worldwide issuance with of course SIM the dominant usage [25]. The other key markets sectors are payment, transport and identity.

Typical SIM EEPROM size in 2015 in Europe was over 256 kilobytes of user memory, but each SIM design has an issuance life cycle of around 3 years. The memory requirement doubles each cycle, as more services are required. In China, a 32-kilobyte-based SIM is very common. As mentioned earlier, the combination of a microcontroller and large NVM memory into one SIM card has been under development for several years. One application is for NFC secure elements either embedded in the phone or in a SIM card. Several millions have been rolled out across Europe in the past few years, and with the advent of schemes like Apple Pay, Samsung Pay and various Mobile Network Operator (MNO) NFC wallets this is going to increase quickly. Mass deployment is dependent on several issues. Operators have to develop attractive and easy to use services. For example with a so-called NFC-SIM mobile operators can provide payment functionality across a number of handset models to subscribers. Alternatively, 3rd-party providers of payment or authentication services can be downloaded Over-The-Air as required by the subscriber. Another drive for NFC-SIM is from operators is to reduce churn of subscribers. By supporting payment on a NFC-SIM, subscribers may want to keep the card when they change handsets, rather than updating their payment service at the same time, so will be more likely to remain with the existing operator. Similarly, the trend by smart phone suppliers to offer payment services, secured by an embedded secure element, encourages customers to stay with the supplier's handsets.

Other key market sectors have their technology drivers based on their business model. Payment cards tend to need a relatively small memory to store a banks issuing certificate, for limited lifetime of three to five years. The specification for most banking cards can be satisfied by a chip with 4-kilobyte EEPROM for basic EMV SDA (Static Digital Authentication) designs, and even with EMV DDA (Dynamic Digital Authentication) 8 kilobytes should be enough, plus the prerequisite support for cryptographic processing. Some multifunctional banking cards have used 16 kilobytes, where using contactless interfaces transport ticketing functions have been included. The driver from the finance market has been like that of the low-end SIM to reduce costs, while maintaining interoperability. This is in unlikely to change. Identity smart cards designs on the other hand have be very diverse, with few standards applying as many have been used for closed user groups. Many early ID cards used little more than the ATR of the chip as an identifier, keeping the users data within the local network. This is now changing with the introduction of the PIV and the ICAO specifications. The driver for identity is not memory but the need to increase security, through enhanced chip design, and convenience in use of the card, with the introduction of contactless interoperable designs.

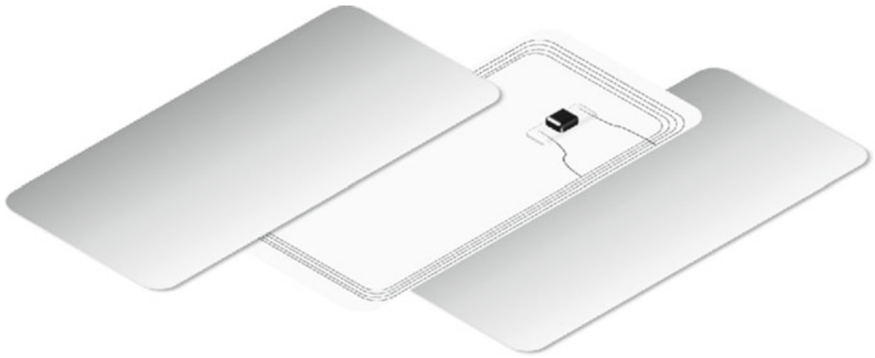


Fig. 15.3 Contactless card—courtesy of Infineon Technologies AG

15.1.3.3 Contactless

As outlined, one technology driver in smart cards is the physical interface to the chip. The original designs were contact chip-based, but this limited the form factor of the card and readers, incurred infrastructure maintenance costs and reduced user convenience. Contactless interfaces were originally developed from work around battery-powered telemetry tokens. Since the early 1990s, contactless, batteryless, so-called passive, cards have been developed. These consist of a chip, mounted in a module, which is attached to a metal wire antenna mounted in a plastic inlay, as shown in Fig. 15.3. The whole assembly is then covered on either side by an extrusion or laminating with layers of plastic, either of a PVC or polycarbonate material.

Today, they have been used in hundreds of applications by millions of people, from very high-volume applications such as prepaid phone cards in Japan to very small volume schemes for single-door access control cards. In the transport industry, 95% of smart ticketing products worldwide currently use low-cost memory cards. The typical data transfer rates of around 100 kbps allow a small number of bytes to be read or written during a transport gate transaction.

The Oyster card™ in London was originally based on a 1-kilobyte contactless memory chip. Similar schemes exist in Seoul and San Paulo. In France, contactless memory paper tickets have been very popular, and in the USA many major cities are or will be using the technology based on contactless banking cards. This trend is likely to continue. Most transport systems are closed operations, i.e. the issuance and acceptance of the cards are via a single operator. The costs of cash handling and opportunities for fraud from paper tickets can be reduced with the use of smart card technology. Many existing transport operators have decided to maintain own contactless card schemes because of the legacy of investments in revenue collection systems, and complex ticket schemes based on time, day, location and service. In some cases, contactless payment cards from banks, based on MasterCard's "PayPass™" or Visa's "payWave™", may erode these schemes, as banks have a wider acceptance by users. Transport operators can also gain from the banks' efficient financial processing capabilities and, not least, negating the cost of their own card issuance and management, but such cards can only support by integrated back office ticketing schemes.

Contactless memory cards can today have sizeable capacity over 32 Kbytes, so the content of such cards can include even biometric data, and extensive transaction records, but the lack of memory management prohibits anything more than a basic function of data fields without the use of external verification tools. Security is limited, like contact memory card, to features such as the unique ID (UID) number of the chip and simple pass code access management. This implies that any sensitive data must be encrypted off chip. The data transfer rates were again small due to power constraints, so typical values are 106 kbps, but there are a few options for faster communications. But the underlying driver for contactless memory card is the assumption that the terminal it is used with is online or has a Secure Access Module (SAM) to verify the UID and content of the device. With such an online system, the chip memory can be reduced to little more than a few bytes. As a result, there are contactless memories with 128 bytes or less, which have been issued in their millions for transport applications mounted in a disposable paper ticket format. In the contactless vending or other remote services like fuel sales, slightly more sophisticated memory chips with a cipher engine to generate a cryptogram have become popular, especially in the US market at fuel stations. Although in principle these memory devices might be open to cloning, the online networks can determine whether a device has been duplicated by the transaction analysis (one card cannot be in two places at the same time) and mitigating any risk by limiting the value of any one transaction.

As microcontrollers have become the technology of choice for contact cards, the desire to have contactless microcontrollers has grown. However, the power requirements of the chip technology have limited the functionality of the card to much lower levels up until the last few years. One of the highest volume programs in Europe has been a contactless transport microcontroller card in France, where ASK have deployed several million using an 8-kilobyte card. Contactless microcontrollers are available with a variety of interfaces; most offer a single standard interface ISO/IEC 14443 type A or B, or like those from Infineon, who offer options for A and B in one chip. There are several other contactless proprietary protocols which have been developed, but very few have survived in the market place. An exception is FeliCa™ from Sony who by working with transport agencies in several countries have supplied several hundred million compatible devices, although this includes memory and microcontroller chips. Controllers which can emulate memory cards have been popular where systems have required an upgrade path. Some such cards can offer dual function options, so that either basic memory fields can be read or Application Protocol Data Unit (APDU) commands and applications can be executed. Contactless high-speed interfaces working at up to 848 kbps have been in circulation since 2007. Similarly, the capability of cards to quickly perform asymmetric cryptographic algorithms while in contactless mode has been a key achievement. Data memory sizes on these cards today are typically for basic city cards around 8 kilobytes today, offering combined transport, payment and city services applications. The introduction of e-Passport has led to a typical demand memory space of 80 kilobytes. In addition, the security functionality and certification required by governments has driven the need for higher performance with low power operation.

In the same way that specific industry sectors have pushed the contact card technology requirement, again it is being repeated for contactless. Future demand is pushing the technology in two directions: low memory options, under 1 kilobytes, and higher memory options with up to 500 kilobytes for multiple biometrics. The financial market across the world is looking to take on other services as described earlier, but with extra convenience of contactless and enhanced security. Within a few years, low-end contactless microcontrollers will have enhanced cryptographic functions with two-factor authentications only required for high-value purchases via one of the two interfaces.

With contactless designs, there are other alternatives with completely different form factors, e.g. the key fob, coin-shaped tokens or in the cover of a mobile phone. More enhanced designs are possible with a Radio Frequency Identification (RFID)-enabled mobile phone, where the data might be stored in the SIM and transacted via a RF link, such as Near-Field Communication (NFC) [10]. This concept is being promoted by the NFC Forum consortium founded by Sony, Philips and Nokia. NFC proximity technology is standardised in ISO/IEC 18092, ISO/IEC 21481, ECMA (340, 352 and 356) and ETSI TS 102 190. It is based around ISO/IEC 14443. It will allow two options: for the phone to act as a token, or as a reader of other compatible tokens. The advantages to the user to download, or upload Over the Air transaction data such as value, or even content, may lead to completely new applications.

15.1.3.4 Dual-Interface Cards

Another driver, legacy support, has led to the combination of contact and contactless interfaces on a card being implemented, as shown in Fig. 15.4. This is with either a single monolithic device with a contact-based module which has internal connections for an RF antenna or as a so-called hybrid card with two independent devices: one

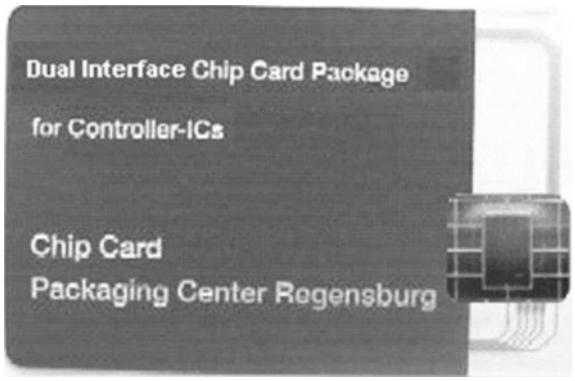


Fig. 15.4 Monolithic dual interface card—courtesy of Infineon Technologies AG

contact, and the other contactless embedded in a card. The most common application is to allow low-value payment to either contactless reader in shop or on public transport without a PIN and contact payment functions with a PIN.

A dual-interface card can all be used for physical and logical access control. This allows the user to identify themselves to a variety of readers and systems with one set of credentials.

A hybrid card can be implemented with a variety of chip combinations, such as a contact chip for the logical access and one or more contactless chips for proximity or vicinity readers in physical access. In this way, the two functions can be tuned to meet the needs of the user and the operator(s) where there is no need for data to be passed between them, for example a banking card and an entry card for city services card and a bus travel card, any reconciliation is done offline.

The dual-interface card with a single chip offers efficiency in issuance and card management. If there is one central authority, e.g. a city council offering a concessionary travel pass combined with kiosk service access for some of its citizens, one card can offer the possibility for the user to interface to a variety of systems. The single chip can be provisioned in a single-step personalisation; this offers faster, lower cost issuance. The single data set on the card provides for quicker consolidation process across different services. In the back office, a single middleware system can be used so reducing transaction processing. The management processes can be simplified with a single chip card, with a single security profile so reducing risks. Finally, the multiple interfaces can improve user convenience.

Dual-interface cards do have issues; if there are multiple scheme operators sharing the card, as to who owns the data and who has the leading relationship to the user? For example, a student card has two applications from two independent issuers, the college and a retail loyalty scheme. If the college revokes the card, when a student finishes a course, the loyalty scheme operator may have extra effort to reconcile the unused loyalty points on the card. In principle, dual-interface cards need common infrastructures behind the readers. Somewhere in the hosting database, the logical access and physical access networks have to be connected. Dual-interface cards may need separate software drivers for each interface, i.e. proprietary messaging protocol for each type of communications maybe considerably different. It has only been a recent innovation where the communications method has been transparent to the card or host. In many physical access control systems, control is often separate from the logical network management. For example, the access card to a building may be under the control of the HR department, but the PC access is an IT management issue. The question with dual-interface cards with multiple applications is often "Whose card is it?". The answer to this administrative question can determine the type of technology used. Existing separate physical and logical access systems may need upgrading if complete compatibility is to be achieved following the introduction of a dual-interface card. In the final analysis, the convenience of contactless can now be extended from simple memory card to complex crypto-controllers with readers capable of reading the all, be it with varying transaction times. With the introduction of such contactless banking cards, and e-passports, the future is contactless without doubt; dual-interface is likely to be a transitional product, much like the audio cassette was between the vinyl LP and the CD.

15.1.4 Technology Trends

In the past, smart card chip designs have changed as silicon technology has improved. The underlying trends seem likely to continue and in some cases accelerate. Smart chip cards have existed for around 30 years, in that time many billions of cards have been issues; in 2017, it is expected that over 14 billion cards will be produced, and based on historical figures the average number of cards issued is predicted to grow by around 5% per year. As the average chip complexity increases, the actual cost per bit is falling; this may not be in direct proportion to Moore's Law [13] but certainly it is closely coupled. If the past twenty years are considered, some strong technology trends are seen: The line geometry, which is the smallest feature size of a chip, has been reduced by over 90%. This means if chips function took 20 mm^2 in 1996, it is now more 99% smaller, i.e. under 0.7 mm^2. The reduced feature size has improved speed and reduced current consumption; new memory designs have improved the size further. A high-end 128 kilobytes EEPROM SIM chip today is around 5 mm^2, which if fabricated in 1996 would have taken several kilowatts of power! The speed of the processor allows fast transactions, using 1996 technology you would have had to wait several minutes to get money from an Automated Teller Machine (ATM). A summary of the changes in features for a typical smart cards from 1996 till 2016 have been outlined in Table 15.1. The features do not reflect any one card, but are guidelines for trends in the market.

The demand for greater performance for lower cost from consumer products, such as computers and mobile phones, has been the economic driver behind the improvement in silicon processing. The smart card trend has followed a different path for several reasons. Firstly, the demand has been distorted by the use of the SIM. If mobile phones did not use the GSM SIM, but relied on network security, as is the case for some alternative systems, then demand for smart cards would have been driven by the payment sector, which needs only low-end security tokens as most of the security is in the online network. The electronic ID sector volumes would not have justified the development of large memory smart cards, and so may well have taken a different path, perhaps using online verification of identity credentials from national databases.

The trends for the future, however, may not directly follow the same lines. Although the cost of a low-end chip may have fallen as silicon costs have reduced and economies of scale have reduced card assembly costs, the demand for more features has meant that average price of a card have been maintained, while at the same time the demand for memory depends on the application, as shown in Fig. 15.5. In some applications, the memory size is likely to remain relatively static; in others as more services are demanded, the memory requirement is likely to grow geometrically.

Table 15.1 Technology changes comparison

	1996	2016	Comment
Geometry	700 nm	65 nm	40 nm will likely be common by 2020
CPU	8 Bit	32 Bit	Moved from CPU to RISC
ROM	20 KB	200 KB	ROM has been replaced in many designs by Flash
RAM (KB)	0.5	8	
EEPROM	8 KB	500 KB	Max in 2016 < 1 MB
Voltage (V)	5	1.5–5	
Speed			
Clock	3.5 MHz	30 MHz	Max 66 MHz with phase locked loops
I/O speed	19.2 kbps	2 Mb/s	Contact
			USB, 848 bbps contactless
Crypto	66 mS	4 mS	512 bit RSA (with Chinese Remainder Theorem)
Security certification	Rare	CC EAL5+	Protection profile changes every 2/3 years

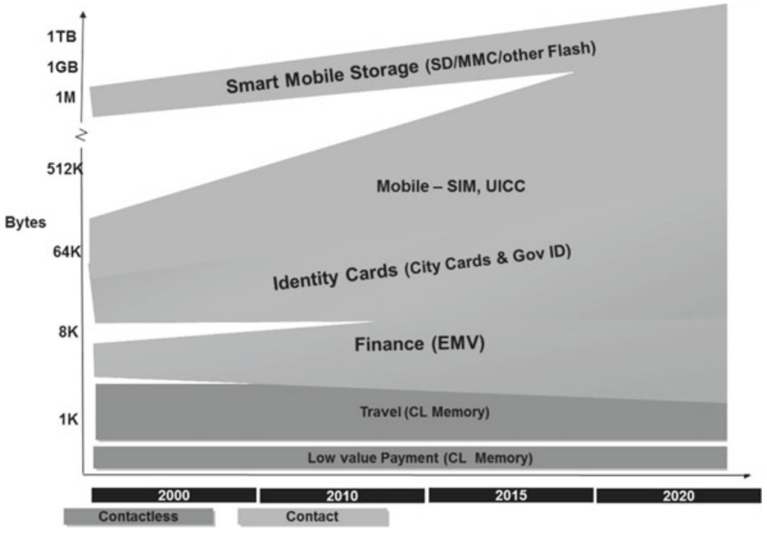


Fig. 15.5 The trends in memory size by application

The underlying design features of chip card ICs are not likely to change for and new applications. Considering the applications in turn, each is constrained by legacy issues which preclude the security functions from being changed quickly, the emphasis will be to use the technology to improve the efficiency of the services.

The financial payment markets are in a process of consolidation. EMV cards are not ubiquitous at present, but 10 years from now may be practically universal. The interface on a Payment Card may change from contact to contactless, but the underlying demand as a certificate carrier and digital signature device will remain relatively unchanged. The demand for higher security will increase. Therefore, the demand will accelerate in the USA further towards contactless, where the pressures of consumer convenience society are sure to have huge effect. The transition to chipped EMV in American payment cards from 2015, which has an existing population around 1 billion magnetic stripe cards, will have major effect on market demand. The payment market will fragment with dual-interface cards to support legacy systems, smartphones/wearables with NFC for low-value transactions and for larger values a contactless microcontroller card with DDA functions, and authentication with a PIN. People already have different cards for different purposes, e.g. personal, business, Internet; it can be extrapolated that the number of payment tokens per person could double in the next 10 years. The underlying chip card IC function and security will not change significantly, so pressure on cost reduction from process improvements will continue.

As stated before, some transport operators are currently moving away from contactless memory cards. This is part of the underlying drive to the cost of physical ticket out of the system; this is not only the ticket itself, but the processes of issuance, payment handling and back office infrastructure. If payment cards can be used for transport ticket transaction, then the transport operators may not want to issue their own cards. Several transport systems are testing the use of electronic tokens stored on mobile phones and read via NFC, again another driver away from operator-specific tickets. As a result, transport smart cards and even paper tickets may follow the way of prepaid public phone cards and become a product which is used only by operators which do not have an online service, or cannot afford the upgrade costs.

The identification market, as outlined before, is dominated by the move to electronic passports and other national programmes. The technology to support the demands of governments to identify users is driven by various security threats, rather than any economic driver. Already in the USA, visitors are required to register 10 fingerprints and a facial image at point of entry. For passport applications, in future, this will follow. It has been foreseen that identity documents might one day also carry these data and include even iris images. Some countries may even allow affiliated states to load entry or exit visas and entry stamps to facilitate speed and improve identification accuracy. These measures have been included in so-called ICAO 9303 LDS 2.0 specification. If implemented, it would require chips with capacity of up to 500 kilobytes or more for personal data, with a high-speed contactless interface. The worldwide trend of government and private requirements to identify individuals by use of smart cards is undoubtedly going to drive demand for other identity cards. Although multifunction or multiapplication technologies exist, and seem convenient for the user, the simple process of a card-issuing authority being able to verify a user's credentials from their own data on the face of a card and in the chip is very strong. However, for one authenticating authority to accept another's format and data, e.g. a library card as an alternative to a transport ticket, presents risks legally, commercially, and is politically a complicated issue. It is probably less risky to issue separate tokens by various issuers. In the UK alone, if employers, local authorities and central government agencies plans come to fruition in the next 5 years, the average adult may have 5 or more such ID tokens, though some maybe virtual, i.e. stored on phone or in a data cloud. What is certain in the security of elements on and in these tokens will increase, as the counterfeiting of identity document activities by criminals and terrorists will not disappear, so designs will have to become more sophisticated in the future and the security of chip card ICs will be a key differentiator.

The mobile phone market, as described, is the largest influence on the chips designed for smart cards. The basic functions have remained essentially unchanged for 10 years. The legacy of several billion phones predicates that SIMs cannot change quickly despite the predicted increase in memory, speed, to support new software and interfaces to support NFC and mobile services. One example is even the latest 4th-generation SIM form factor announced in 2012 which is just the chip module embedded in a plastic housing 12 mm by 9 mm, it is delivered in a standard credit card (ID1) size format and be pressed out during a phone's initial set up by a subscriber. This format reduces the space taken up in the phone so allowing for smaller designs; this format is little more than the chip module embedded in plastic. There have been some moves to embed a chip in a surface mount assembly on the motherboard of the phone. This reduces the cost of handling of the SIM, which is a complex logistics operation and but requires the SIM (embedded Universal Integrated Circuit Card—eUICC) can be reassigned to work with multiple network operators. The long-term trend will depend on consumer fashion as much as cost. The mobile phone is quintessentially driven by convenience. The SIM will survive as it becomes the focus of the user's rights management for content in the user's environment rather than just a security function for the operator. How then to ensure the data integrity in case of loss or theft will become a crucial service for the operator and the user.

Over the years, various complex assembly smart cards with more than a basic data processing function have been developed, though until now few have made it past the prototype stage, with production volumes numbered at best in the thousands. These cards are much like concept cars and have been designed to satisfy either a bespoke user group or to show off new technologies. The integration of display technology from Liquid Crystal Display (LCD) to Organic Light-Emitting Diode (OLED) has been tried. The ability to display on the surface of the card details such as a value of a transaction, an authentication cryptogram or perhaps the balance of points is superficially attractive. However, the mechanical stress exerted on such a display when a card is flexed in use is an issue. Similarly, the power requirements of the display means the display can often only be used when placed on or in a reader. Should a technology which offers very low-cost assembly and ability to withstand the bending and torsion stress inside a wallet or purse become available, this might change. A more likely use case is if the card holder can read and interact with a card

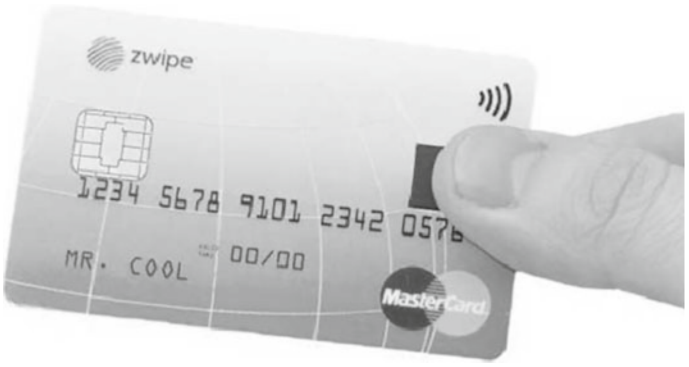


Fig. 15.6 Fingerprint activated payment card—courtesy of Zwipe Norway

via NFC on their phone. This then only requires the security of any data transferred to be preserved, a simple software process rather than a complex mechanical design. One concept which has generated attention has been the process of card holder authentication. The psychological attraction of verifying literally who has hold of a card by the use of an integrated function such as a PIN pad or a fingerprint biometric sensor is very strong. Adding a PIN pad has been tried but since some users write or scratch their PIN on to the back of the card, this is not considered a secure feature. The inclusion of fingerprint sensors, even with contactless interface, as shown in Fig. 15.6, has been under development for some time. The key issue is mechanical, that is to be able to withstand the daily use a sensor must be able to withstand the card being flexed. The sensors, around 10 mm by 10 mm, have been encased in a way so as to reduce the stress on the chip, but this adds extensive costs to the card.

As with the display concept, if fingerprint sensors in phones become popular for access control, it would be easy to extend the feature to require a user to have their card/phone and their finger on the phones sensor. Other biometric recognition processes such as the users' voice, facial features and even iris could also be supported in this way. Verifying and even loading a card with this value has been demonstrated, but whether there is a real-life business case is unclear.

Another function for inclusion in a complex card assembly has been the addition of power source into the card. For contactless cards, the limits of physics have precluded data being processed beyond the range of the induction power of a reader. Data reading can be further, perhaps up to one or two metres, for simple memory devices. With the addition of very thin batteries, a more powerful processor can be embedded, with perhaps added functions such as displays. Powered cards are more of a telemetry function rather than a smart card for secure applications, but there might be an overlap in the future where valuable mobile objects need to have their integrity managed remotely such as shipping containers, cash transit boxes or even vulnerable people.

Finally in the area of complex card assemblies, there are several concepts for cards which serve a very specific need. SIM cards with other integrated wireless chips such as Global Positioning System (GPS), Bluetooth Low Energy (BLE) beacons or Wi-Fi have been demonstrated, but the business case for such a product has yet to be developed.

Long-term future developments for smart cards are dependent on the balance of commercial justification, the availability of technology and the acceptance and demand from the card user. The concept for one card to hold every application that a user might need in day-to-day life is technically feasible today. The commercial justification is difficult because of the diverse interests of the various application stakeholders. Some users might find the concept attractive, but others are concerned by the thought of the possible loss of all their personal data if such a card is lost. Currently, many people store on their smartphones a lot of personal data, backed up onto cloud-based server. The use of tokenisation of payment credentials could be extended to identification data if required, till now the function of the SIM embedded or otherwise is limited to that of mobile services. However, across the world, some MNO's, for example, in China have been working with major payment providers to issue large volumes of NFC SIMs to store payment, transport ticketing and other personal credentials. Smart cards with large number of functions would still require a cloud-based backup process. There is the alternative perspective that as the world becomes even more networked, by the use of a federated identity, a card or token could be reduced to holding a single credential, a sort of single sign-on function to the digital world. These concepts, such as Microsoft Hello, are attractive for users needing to access a multitude of applications, but there is again privacy concerns that should such as system be hacked, then a user's credential could be misused and either counterfeit or cloned credentials could take a very long time to cleanse. The most commonly held view is that the number of single function cards will continue to multiply.

15.1.5 Emerging Applications

There is a growing trend to use chip card IC technology in form factors other than a card. These can be grouped in two generic areas: embedded, where the security function is still a discrete device but is connected to other non-secure devices; and integrated, where the security function is part of a larger device. These two approaches have similar advantages and disadvantages irrespective of the actual application. The underlying driver is the business case and is balance of the function, cost, security and data management: each of these elements have to be considered in detail before a decision is made as to what format is used. One of the most highly discussed markets is for IoT security. But since IoT is generic term covering a myriad of applications, the security functions used are equally diverse. Many IoT products and services have little or no security, even in software, let alone hardware.

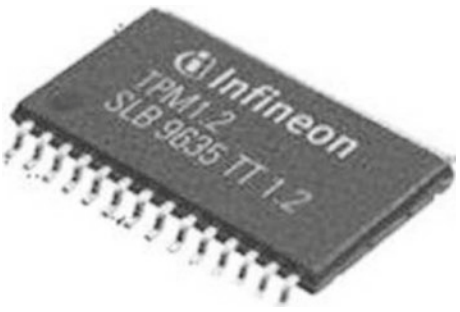


Fig. 15.7 TPM—a surface mountable smart card IC

15.1.5.1 Embedded Security ICs

Embedding of a chip card IC normally takes the first step of repackaging the chip into a circuit board assembly. This can be done by wire bonding the chip by first assembling the chip in a standard plastic IC housing, as shown in Fig. 15.7. This has the advantage of preserving the security integral to the design of the chip card IC and of course making it easier to assemble. It can require the chip to be modified to provide an electrical interface which can be used with the other chips on the board such as the CPU or Digital Signal Processor (DSP). A chip card IC normally communicates using the ISO/IEC 7816 serial protocol. Most multichip logic designs can provide a serial interface which can be matched to the chip card IC, but this often requires some software overhead to match the data framing and timing requirements. Some chip card IC designs have had standard serial interfaces added such as Universal Asynchronous Receiver/Transmitter (UART) functions added, Serial Peripheral Interface (SPI), Inter-integrated Circuit (I2C) or USB. In some cases where the volume justifies the cost, a specific interface for the application has been developed. This is the case for the Trusted Platform Module (TPM) used in PCs and industrial IoT systems as a root of trust for trusted computing designs [8].

Cost is of course the main driver; any change to a chip card chip may require the device to be re-certified by an approved security evaluator, and this can add time and further cost. Another issue is the area that the security device and its connections take up in the overall assembly. In products such as the mobile phone, space is at a premium. Finally, the embedded device needs managing during its life cycle: at the beginning when it is to be initialised, this may be for the injection or generation of cryptographic key pairs and the personalisation of the chip; during normal operation when updates maybe required in the field; and when the product is disposed of, to ensure security data are either protected or erased. This life cycle management is specific to the application, but if the complete product integrity is to remain unblemished, then a security profile needs to be defined at each stage to prevent attack.

There are several publicly known systems of secure embedded designs which have been successfully deployed. Probably one of the most earliest was in the pay-TV set-top boxes used in various broadcast systems from terrestrial, cable and satellite receivers. The copyright protection requirements of the systems has employed various digital decryption designs based around either Application-Specific Integrated Circuit (ASIC) or full custom ICs which use fast data processing to rearrange the video and audio TV signals. The decryption of the data is sometimes done inside a Personal Computer Memory Card International Association (PCMCIA) card, a so-called conditional access module. In many designs, the key for the deciphering is kept on a smart card; this is partially to ensure that the broadcaster has some control via the user of the rights management, and partially to secure the access control from potential attacks by people wanting to steal the content. For many cable systems where the box is constantly in a two-way link to the network, the security IC is embedded in the box and can be verified easily. The cable box remains at all time the property of the cable operator, and with the security and the identity built into the box can only be installed with the operators consent. With a simple unique identity for the box the integrity of the connection can be authenticated and the content can be decrypted with keys downloaded from the host server. The keys maybe kept internally to the baseband decryption devices or on a PMCIA module. The variety of security architectures is partially a commercial issue, to ensure that systems supplied by one broadcaster cannot be used for another, or even the same broadcaster in a different country or region. Current IP-related solutions like Chromecast, Apple TV, employ software security-based whole network architecture. Streaming services like Netflix use one-time secure IP session keys to reduce the ability to copy the content.

In the past decade, embedded security in cars and other automotive vehicles has increased. The keyless entry systems used for door access and engine immobilizers have become almost ubiquitous. The RFID link between the car and key uses one of the standard 400 MHz UHF bands for telemetry. Originally, the security protocol used data coding techniques were based on a fixed code sequence. This was easily cloned by eavesdroppers using radio scanners, who captured the code and rebroadcast it later. Then, so-called "rolling codes" and simple encryption to stop this attack were developed and more recently codes which dynamically changed during use. The chip security has been minimal, relying on the key being kept physically secure as a sufficient deterrent. The motivation for this extra security was driven by insurance companies to reduce the consequences of theft. Its introduction was originally discussed in the late 1980s but was considered too expensive for most cars. The simple process of offering a discounted insurance premium, on vehicles with extra security, brought an almost overnight revolution, and literally drove the use of embedded security. The problem remains moving target as has been shown in the Megamos keyless entry attack involving the theft of hundred if not thousands of valuable high-end cars [26]. However, the increasingly sophisticated hacking community is requiring automotive systems to use ever more secure silicon to protect the "connected car". After the well-publicised 2015 wireless attack on a Fiat Chrysler Jeep Cherokee, subsequent mitigation has been an expensive wake-up call for the industry.

An interesting example of embedded security is the new digital tachograph that has been introduced across Europe. This design uses a combination of security processes to ensure the data recording a driver's behaviour complies with local and European laws on time and speed is held securely. The driver's identity and most recent driving information are held on a smart card. The data recording the behaviour of all drivers of a vehicle for the past year are held encrypted in the vehicle unit in the cabin. The data in the unit are encrypted by an embedded device in the cabin mounted vehicle unit and can only be read by an inspection authority, or police, who have their own smart card. Finally, the vehicle data are generated from the engine of the vehicle, via a special sensor which picks up rotation information and sends it encrypted to the vehicle unit. The sensor design is a special custom design based on chip card IC security principles but engineered to survive in the environment of a gearbox for the life of the vehicle. So there is a chain of security from the machine to the terminal to the user, and extendable to any inspection system. It has been considered to use similar technology for private vehicle applications such as road pricing, but this is one of many possible solutions. Automotive security has been further enhanced with embedded security devices in engine management systems so ensure the vehicles software can only be adjusted by authenticated service technicians. It is conceivable that a vehicle could be geo-fenced to allow only a certain performance in a specific area, or a period of time or for a particular user, might be managed with advanced chip card IC technology, so that the car use becomes a paid for service. At present, several insurance companies have used GPS assisted M2M GSM technology to ensure that a car is used only at certain times, with speed limits and within a known region, but it does not link directly to the driver. In the future, the driver and their car will be securely connected to a cloud based server at all times.

15.1.5.2 Embedded Financial Security

Another use of embedded security is in point of sale systems where a terminal's authentication keys are held on a Secure Access Module (SAM). Most commonly the SAM takes the form of a SIM-shaped token and may be a microprocessor smart card with a large NVM used to store the private keys of the card scheme operators or transaction acquirer; this may be a bank or similar institution in the case of transport authority. Terminals may have space for several SAMs if the unit is to be used with several schemes. In some systems, individual SAMs have been replaced by server-based systems offering lower costs to terminal operators.

15.1.5.3 Infosecurity Technology

As software is a commercial product in its own right, software and content suppliers have wanted to ensure some sort of rights management on the use of their intellectual property on the hardware of their choice or use by only authorised users, in some cases software licensees. To ensure the integrity originally simple logic designs manipulated signals or data, this modified data were taken as sign of authenticity by the software on the host computer. The logic was encapsulated in a discrete module plugged into a communications port of the host in the form of a bespoke hardware token "dongle". When the dongle is not present, the software runs in a restricted mode or refuses to run. Dongles are used by some vendors as a form of copy prevention or digital rights management because it is much harder to copy the dongle than to copy the software. The original devices were used with minicomputers and later games machines in the 1970s; their mere presence was a deterrent. With the advent of the PC, software became even more transportable; the result was that dongles were worth counterfeiting. Dongles then started to incorporate increasing security features. Modern dongles use chip card IC technology with both key material and even software applets encrypted into the device. The most common form factor is in the form of a USB key, but there are many variants. Some are simple mechanical devices which allow a SIM card to be inserted. Some designs include some form of two-factor authentication, where a display on the dongle must be read every time it is used, with such a cryptogram can generated as a form of one-time password. Two-factor authentications have been used, where a fingerprint sensor has been incorporated on to the device. But if the cryptographic information provided by the dongle can be cracked or hacked, then the software or service can be pirated. The only solution is to embed security into an otherwise open platform, and allow it to be switched on when required. Such as solution is the TPM of a laptop PC and in future any device which uses open platform, e.g. smart phone, may have embedded a security chip, in the form of a "Trusted Computing" device. Essentially, this is a chip card IC with a special serial interface suitable for the host architecture, and it provides a root store for a set of private keys, which have been set up by a known trusted source. The result is that software rights for the platform can be determined and checked every time it is switched on. Depending on the use model, these rights can be controlled by the software owner, or the user. With the addition of a smart card or secure dongle, a whole software rights environment can be defined where the integrity of the platform and the identity of the user can be validated. In a corporate environment, it places control in the hands of the network administration to determine what hardware and software are allowed on a network. One of the first large commercial deployments of trusted computing was with Microsoft Windows. Since the introduction of Vista and versions since have offered have a utility for disc encryption called BitLocker. Drive encryption protects data by preventing unauthorised users from breaking Windows file and system protection on lost, stolen or inappropriately decommissioned computers. This protection is achieved by encrypting the entire Windows volume; with BitLocker, all user and system files are encrypted including the swap and hibernation files. Integrity checking the software components helps to ensure that data decryption is performed only if those components appear unmolested and that the encrypted data is located in the original computer. Other operating systems such as Google Chromebook's Chrome OS rely on TPMs.

Other potential applications for embedded secure elements will increase as the number of devices becomes connected as part of the movement to the Internet of Things. In the home, for instance, there may be several requirements. Although smart meters and Internet-connected heating systems currently rely on a secure software network connection, some may start to include hardware security if threats become more prevalent. In another decade, it is foreseeable that users may have more secure chip card ICs in embedded electronic homes and cars than they carry smart cards on their person today.

15.1.5.4 Integrated Security IC Technology

The underlying drivers of increased threats to intellectual property and sophisticated hacking of devices have generated another demand for security IC technology, to protect a design by integration into another IC. The prime market is for consumer multimedia equipment, such as smartphones, games systems or other devices which need to secure software content or service integrity, especially where content is provided by a third party with extra concerns over liability. Certainly, the security of these platforms has been an increased as the trend to open systems allows for new cyberattacks. A single chip device has the advantage of reducing the costs and handling of a separate security device. An example is the smart phone's media functions which are generally located in a specific media processing chip, the "applications processor", but there is a tendency to integrate this device into a single chip device handling all digital functions. If such a device was compromised, then the content could be manipulated. Solutions to include some form of security function are being developed; the choice is based on economic and security issues. From an economic point of view, the cost of the device is only part of the equation, the cost of ownership through the entire life cycle of the product and its content can be a dominant factor. If a product, as with many consumer designs, has a short lifetime, then these costs have to be amortised quickly. One proposal has been to provide a Trusted Execution Environment (TEE) as a portioned function is the main processor. Although a TEE can provide some software-based security, as it is produced on the same silicon with the same manufacturing processes the TEE is not protected against invasive or observation attacks on a specific chip. Such a fully integrated chip cannot be designed to meet the protection profile of a chip card IC, not only because of cost, but the influence of data processing performance. However, the TEE can provide a measure of protection. When used in combination with a secure element, it can provide a useful level of security.

In the case of games machines, this hardware cost has in the past been offset against revenue from the games software over several years. There is a dilemma where suppliers want to offer freedom of choice to their users while protecting intellectual property. With relatively open game systems, there has been a growth in suppliers of games machines add-on boosters, "cheat boxes" etc. With closed proprietary designs, the ability to upgrade software or add-on features has often been limited, resulting in a short product life. The trend has been towards online streamed games, ensuring the content is never fully in the user's device and so protecting the intellectual property. Streamed games may give users broadband bandwidth problems and limited local processor performance.

15.2 Conclusions

In brief, the factors which have driven chip card IC technology in the past thirty years are likely to be same for at least the next ten years. These trends for smart cards can be summarised in a simplified roadmap made up of four elements: technology, functionality, form factors and not least security, as shown in Fig. 15.8.

The physical limits on the silicon used will be pushed further; the trend in average memory size is likely to accelerate rapidly as multimegabyte e-UICCs are issued. These will be is based on innovative secure Flash memory designs using two bits/transistor concepts so making the memory even more dense and cheaper per bit. To accommodate the demand for more content, faster interfaces will become standard, for contact-based designs the T = 0 serial interface may become the RS232 of the smart card world, supported but only used for legacy systems, USB may become the contact interface of choice. As logic designs use silicon geometries lower than 50 nm, clock speeds could pass 100 MHz if required, and the power consumption per bit will fall proportionately. The trend to contactless designs running over fast interfaces will grow; as Very High Bit Rate (VHBR) becomes more available with speeds in the range of several megabits/second, then in ten years, it is feasible that contactless will be the interface of choice. The functionality as result will be become very diverse as applications drive smart card technology in different directions. The result will be a number of form factors besides the basic card format, although because of individual

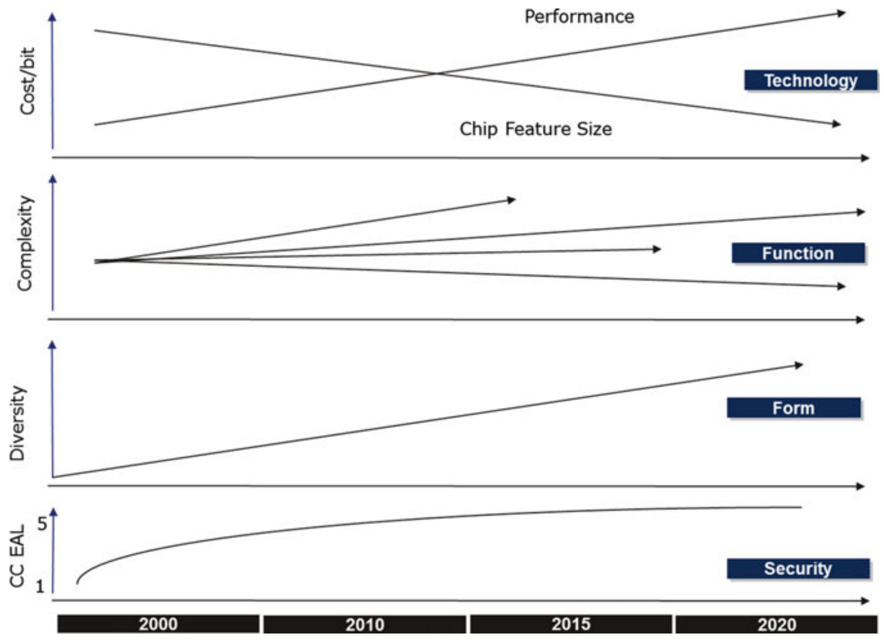


Fig. 15.8 A roadmap of current trends in smart cards

issuers needs the card is likely to remain the dominant visible form for many years to come. The demands for integrated and embedded functions will dictate the dominant form factors in many other instances. The security of products has a limit in terms of the current Common Criteria accreditation process. The highest currently is EAL6+high, which uses some features used in the more formal EAL7 criteria. Most current smart cards do not have a high-security evaluation; many applications do not need it. But more applications will require a formal security assessment. The protection profiles will continue evolve, but it is likely the overall the assessment demand will plateau, and the focus will move the security of the complete system. The underlying trend that drives the mass acceptance of designs and reduces costs is interoperability. It is unlikely there will be one card edge specification, more likely the plethora of acronyms such as CAC, EMV, GSM, ICAO, etc. is likely to grow, but it will be relatively few compared with the thousands of proprietary specifications which exist today.

Above all, the demand for increased security is certain to intensify. Government-backed projects will influence many designs and will impose security guidelines on a wide range of applications which interact with government systems. The trend for falling costs not just in silicon but for all aspects of chip card IC design will continue. Card scheme operators will spend less on developing new bespoke designs but buy more standardised designs with the confidence that they will be fast, interoperable and secure.

™: Trademark: all products, services and licences marked thus are copyright of their respective owners

References

1. A set of specifications are publicly available http://csrc.nist.gov/piv-project, Cited 5 May 2016.
2. Bundesamt für Sicherheit in der Informationstechnik, “Security IC Platform Protection Profile with Augmentation Packages” Certificate BSI-PP-0084-2014.
3. CAC info http://www.cac.mil/ CAC Case study whitepaper by Smart Card Alliance 2001: http://www.smartcardalliance.org/resources/lib/DoD_CAC_Profile.pdf Cited 21.9.2015, US official information https://www.cac.mil/Home.do
4. Copies of the EMV specifications https://www.emvco.com/specifications.aspx, Cited 05 May 2015.
5. Council Regulation (EC) No 2252/2004 of 13 December 2004, http://ec.europa.eu/dgs/home-affairs/what-we-do/policies/borders-and-visas/visa-policy/index_en.htm, Cited 5 May 2016.
6. European Union Directive establishing the framework for electronic signatures: “ Directive 1999/93/EC of the European Parliament and of the Council of 13 December 1999 on a Community framework for electronic signatures. and ”Commission Decision 2003/511/EC adopting three CEN Workshop Agreements as technical standards presumed to be in accordance with the Directive
7. UK card statistics, http://www.theukcardsassociation.org.uk/wm_documents/UK%20Card%20Payments%202015%20Summary.pdf
8. Root of trust white paper June 2012 SANS Institute, http://www.trustedcomputinggroup.org/resources/implementing_hardware_roots_of_trust, Cited 5 May 2016.
9. EU Tachograph website with specifications [5], Cited 5 May 2015.

10. Further information on NFC can be found on the NFC Forum website via www.nfc-forum.org, Cited 5 May 2016.
11. ISO/IEC 24727-1:2007 PDF version (en) can be found on the ISO website www.iso.org, Cited 5 May 2016.
12. Mobile Operators Association figures: http://www.mobilemastinfo.com/stats-and-facts/, Cited 18 Sep 2015.
13. Moore's Law is the empirical observation made in 1965 that the number of transistors on an integrated circuit for minimum component cost doubles every 18 months. It is attributed to Gordon E. Moore a co-founder of Intel.
14. Standardisation process information via http://www.etsi.org/, Cited 18 Sep 2015.
15. The Common Criteria Organisation, "Common Criteria for Information Technology Security Evaluation V3.1", 2012. Source: http://www.commoncriteriaportal.org/cc/, Cited 5 May 2016.
16. National Institute of Standards and Technology http://csrc.ncsl.nist.gov/ has published FIPS 140-1 http://csrc.nist.gov/groups/STM/cmvp/standards.html "Security Requirements for Cryptographic Modules" concerns physical security of smart card ICs, Cited 5 May 2016.
17. ICAO 9303 passport specifications, http://www.icao.int/Security/mrtd/Pages/Document9303.aspx, Cited 5 May 2016.
18. The JavaCard specification can be licensed from Oracle Inc, http://www.oracle.com/technetwork/java/embedded/javacard/downloads/default-1970005.html and its implementation is managed by Global Platform http://globalplatform.org/, cited 21 Sept 2015.
19. The specification in ten parts is available for download from http://www.itso.org.uk/the-specification/, Cited 5 May 2016.
20. The Uniting and Strengthening America by Providing Appropriate Tools Required to Intercept and Obstruct Terrorism Act of 2001 "USA PATRIOT Act", http://thomas.loc.gov/cgi-bin/bdquery/z?d107:HR03162:%7CTOM:/bss/d107query.html%7C, Cited 5 May 2016.
21. FFA UK Annual Review 2015 http://www.financialfraudaction.org, Sept 2015, Cited 5 May 2016.
22. GMSA subscriber growth, http://www.gsma.com/newsroom/press-release/one-billion-new-unique-mobile-subscribers-by-2020-gsma/, Cited 18 Sep 2015
23. EN 15320 Identification Card Systems - Surface Transport Applications - Interoperable Public Transport Application (IOPTA), http://www.cenorm.be/CENORM/BusinessDomains/ Cited 2 Apr 2006.
24. Secure Flash memory, https://silicontrust.wordpress.com/2011/04/06/secure-flash-for-high-security-smart-cards/, Cited 5 May 2016.
25. Smart card statistics, http://www.eurosmart.com/publications, Cited 5 May 2016.
26. Megamos crypto attack, http://www.computerworld.com/article/2971826/cybercrime-hacking/hack-to-steal-cars-with-keyless-ignition-volkswagen-spent-2-years-hiding-flaw.html, Cited 5 May 2016.
27. Smart card statistics derived from Eurosmart data, http://www.eurosmart.com, Cited 5 May 2016.

Chapter 16
Securing the Internet of Things

Paul Dorey

Abstract The *Internet of Things* (IoT) is an umbrella term used to describe the rapidly growing population of smart and/or embedded digital devices that can leverage, monitor and control systems and components in the physical world. In order to understand the need to safely maximise the many potential benefits of these important technology developments, this chapter describes the security challenges that they bring and how these challenges arise. Based on past experiences of security in IT and Industrial Control Systems, as well as the special characteristics of the IoT, the author identifies the need for security work by those working on IoT systems in the three key areas as follows:

- Establishing mechanisms to determine a firm's direction and stance for both security and privacy risk/leadership to inform its IoT deployment and use.
- Making sure that a robust risk assessment and security framework is held up against proposed IoT designs/deployments.
- Developing a clear position on whether to use open versus closed IoT systems, whilst recognising that the purpose of many IoT systems will expand and evolve substantially over time.

In this chapter, the means for addressing these three challenges are combined into a proposed overall decision-making framework for developing an IoT security strategy.

Keywords Internet of Things · IoT · Wearable · Mobile · Mesh · Privacy · Industrial control · ICS · Industrial internet · Cloud · Agency · Big data · Security by design · Strategy · Device · Embedded system

P. Dorey (✉)
Information Security Group, Royal Holloway, University of London,
Egham, Surrey TW20 0EX, UK
e-mail: paul.dorey@csoconfidential.com

K. Mayes and K. Markantonakis (eds.), *Smart Cards, Tokens, Security and Applications*, DOI 10.1007/978-3-319-50500-8_16

16.1 Introduction—What Is ‘The Internet of Things’?

The term ‘Internet of Things’ (IoT) correctly catches the eye of the media as being something very different from standard IT that is creating a new role for technology—which the networking company Cisco has even called ‘The Internet of Everything’ [1]. Perhaps the best definition comes from the International Telecommunications Union paper Y.2060 [2] which defines the IoT as a ‘global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies’. ITU-T Y.2060 also provides the following definitions:

Device: ‘a piece of equipment with the mandatory capabilities of communication and the optional capabilities of sensing, actuation, data capture, data storage and data processing’.

Thing: ‘an object of the physical world (physical things) or the information world (virtual things), which is capable of being identified and integrated into communication networks’.

This description of technology may seem like just more of the IT systems we have become used to, but the physical proximity, sensing and physical actuation capabilities of IoT introduce a whole set of new capabilities that introduce new risks and security challenges.

There is great commercial interest in the IoT because these new innovative devices and embedded *smart* capabilities enable the transformation of the customer experience, enhanced functionality, improved product maintenance and the creation of new business propositions such as subscription models and new forms of data analytics. As a result of the benefits, sweeping adoption is predicted with forecasters variously anticipating between 25bn [3]—50bn [4] IoT devices by 2020.

As suggested in Table 16.1, where the devices appear and, more importantly, the context in which they operate varies considerably, which is illustrated by the following examples of categories:

Industrial Internet of things The move of standard IT systems into the industrial control environment has been happening for the past 20 years. Computer systems that monitor and control industrial processes (such as measuring temperatures and opening valves or adjusting power grids) initially started out as

Table 16.1 Internet of Things estimated millions of units installed by category. Source: Gartner (November 2014)

Category	2013	2014	2015	2020
Automotive	96.0	189.6	372.3	3,511.11
Consumer	1,842.1	2,244.5	2,874.9	13,172.5
Generic business	395.2	479.4	623.9	5,158.6
Vertical business	698.7	836.5	1,009.4	3,164.4
Grand Total	3,032.0	3,750.0	4,880.6	25,006.6

proprietary systems but have migrated to those that use common IT protocols using Ethernet networks, and controllers and sensors communicate to operator consoles that are often Standard Windows PCs. Disruption of these Operational technology/Industrial Control Systems (often over-simply referred to as 'Supervisory Control and Data Acquisition—SCADA' systems) could in some cases cause physical damage or breakdown of processes and as a result have been the subject of a catch-up programme of security improvements. The Industrial IoT (IIoT) is an extension of these systems, usually adding many more distributed sensors and instrumentation of previously passive components.

Examples include monitoring rotating equipment (such as jet engines) for vibrations to allow prediction of failures and preventative maintenance, or real-time sensing of pressure, temperature and chemical composition of the full depth of an oil well.

Building Management Systems BMS' are a special case of Industrial control Systems which are used to maintain the safety and comfort of the occupants of a building by operating Heating, Ventilation and Air Conditioning (HVAC), fire protection, security and supply of water and electricity. These systems often connect with existing IT networks and can impact on the security of those networks.

Home Home IoT appears where all manner of devices in the home, from TV entertainment systems, through to lighting, heating and security may be connected to and managed by services on the Internet. This can be a very mixed environment from smart meters managed and provided by the utility provider through to consumer purchases of low cost sensors and smart light switches. This consumer category of IoT is seen as the greatest growth potential and is already creating unexpected use cases where passive devices such as refrigerators and door locks can be given network connectivity. Even home routers, printers and Wi-fi boxes could also be considered to be home-based IoT devices.

Automotive IoT deployment by the automotive industry is happening where the already highly digitally enabled vehicle is now gaining connectivity to the Internet and the ability to be linked to other systems such as mobile phones, as well as the new potential of remote control or autonomous driving.

Wearables IoT includes wearable devices that include watches, smart jewellery and fitness tracking sensors, through to devices that can have a medical application.

Healthcare IoT in healthcare can provide real-time monitoring of patients, by bedside devices and wearables but also the opportunity for drug delivery by embedded devices, or computerised electrical stimulation such as computer-assisted heart pacemakers and even neuro stimulation of skeletal muscles or the brain itself.

Retail Retail applications include sensors detecting customer interaction with shop displays and proximity devices which are able to communicate with nearby mobile phones.

Agriculture Some consider that agriculture could be the biggest potential growth area for IoT devices that include identification tags on animals able to transmit more data on health and welfare, as well as numerous sensors locally monitoring groups of plants for light, water, fertiliser levels and even a count of pests.

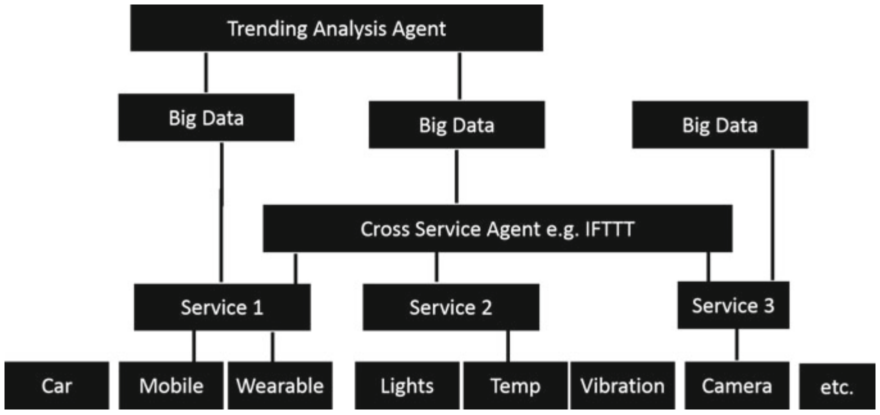


Fig. 16.1 IoT Systems are more than just the device, including services, data stores and applications

In all of these business applications, the IoT system will comprise a whole set of applications and data stores, as well as devices. Applications may also interact with one another and combine through service brokers (See Fig. 16.1).

16.2 The IoT Security Risks

Just as with IT systems, the impact of a security attack on an IoT system is directly related to its purpose and the criticality of the information or processes that the system is handling. The consequences of a security breach of a device that is closely connected to a physical system or an individual therefore reflect that situation and could involve impacts such as:

- Loss of privacy through unauthorised monitoring by cameras, microphones or of other data channels.
- Breach of physical security.
- Loss of control—such as someone maliciously taking over an automobile, interfering with environmental controls, or attempting to cause a spillage or explosion.
- Service unreliability—such as interfering with logistics or electricity/pipelines.
- Product corruption—such as caused by affecting the quality of a manufacturing process.
- Providing false or unreliable information—such as manipulating a medical system or application, or fraudulently manipulating supply metering to cause undercharging.
- Taking control of an IoT network so as to gain 'backdoor' entry into an attached IT network or other network.
- Taking over IoT devices to use them to attack others on the same network or the internet.

16.2.1 Early Security Experiences

At the time of writing,[1] security researchers [20] and the media have been having a field day identifying IoT security weaknesses—not just in the lab, but in devices that are in widespread use. Recent examples include the following:

- TV home entertainment systems that have turned on built-in cameras and microphones without customer permission, or as part of overly wide (and generally unread) terms of use agreements.
- Cars, light bulbs, door locks and refrigerators that have been remotely controlled by attackers, or infected by malware so as to become part of a 'botnet'.
- Studies by both commercial and university research groups that have catalogued numerous IoT software weaknesses, and shown that many phone apps (including some popular medical apps) have been poorly designed and risk exposing sensitive customer data.

These security weaknesses have come about in a number of ways, and devices have been found to contain unpatched software vulnerabilities, poor authentication (including hard coded administrative passwords) or carry out unencrypted transmission of security credentials or sensitive data.

The lack of properly designed remote security maintenance/security patch updating processes seems to be a particular problem. Once deployed it may not be practical to gain physical access to the device to do software transfers by laptop or USB stick, and even if possible, it may still present security challenges. A particular illustration in 2015 [5] was the complete remote takeover of a car, including the steering, brakes and acceleration. The software hack could be done by the researchers over the air, but the car manufacturer had to recall vehicles to their dealerships to apply the security patch. After a series of complaints due to the inconvenience and time this recall would take, the manufacturer then took the decision to post out USB sticks to their car owners so they could do the update themselves. Some security commentators were very critical of this, saying that a new attack vector was now available if car software updates were distributed like this. It is not clear whether or not the software updates to all cars have any form of signing to show their integrity and authenticity.

A further challenge is introduced by the use of mobile devices which may form part of the user interaction with the IoT environment. We have seen some improvement in the design of mobile apps but, at the time of writing, researchers are still finding development and design mistakes such as (again) the unencrypted transmission of credentials and sensitive data, or inappropriate use of shared device logs or storage areas. Some researchers have demonstrated the ability to capture security credentials from a public Wi-fi connection and then use these to unlock or control a house with its own IoT systems.

The OWASP Internet of Things Project [6] has documented the Top 10 IoT vulnerabilities collected by project members, and these are found to be in a whole range of places:

[1]May 2016.

1. Insecure web interfaces
2. Insufficient Authentication/Authorisation
3. Insecure Network Services
4. Lack of Transport Encryption
5. Privacy Concerns
6. Insecure Cloud Interface
7. Insecure Mobile Interface
8. Insufficient Security Configurability
9. Insecure Software/Firmware
10. Poor Physical Security (including local USB access)

16.3 Why Are These Security Problems Happening?

The obvious question, considering the previous decades of work to improve information security and security awareness in IT, is why relatively basic security mistakes are being made in the new IoT environment? Of course security mistakes in IT still happen, but there are some particular characteristics of the IoT environment which are contributing to the problem:

16.3.1 Skills and Knowledge Gap

Firstly, we see the environment demonstrating a lack of understanding of security issues and lack of expertise in implementing secure systems by both system designers and implementers. Many IoT systems and components are coming from a development community of electronic, electrical and mechanical engineers who have been used to closed systems with limited, and closed, networking capability and relatively simple software implementations.

The industries using IoT are also now expanding to including those which have had limited exposure to IT systems at scale and so have seen nothing like the cyber attacks experienced by banks, governments and IT service providers. The assumption has been that (for example) agriculture, transportation, healthcare and even retail sectors have been ‘below the attack radar’ and so these businesses have not made the investment in developing security skills and capabilities. Also few sectors, unlike banks and telecommunications companies, have the experience of securely operating highly distributed networks and dealing with protecting vast volumes of data. Both of these challenges—of technology outside of traditional IT platforms and addressing big scale and big data—come with adopting and deploying IoT.

This knowledge and communication gap between traditional engineers and IT security professionals was first experienced around 2000 in the process industries, such as oil and gas and the utility sector. My personal experience as a Chief Information Security Officer for a major global oil and gas company was to discover that (in 2001) my IT-trained security team did not understand Industrial Control Systems (ICS), and the industrial control engineers had no formal background in information security. The security challenges were compounded when we found that the traditional control systems had not been designed with digital security in mind, and there had been a steady adoption of standard IT systems and protocols within the ICS environment. Since this time, the security of industrial control systems have advanced considerably, and the engineers designing new systems are typically taking security into account. Engineering publications such those from the IEEE are also addressing information security challenges and solutions and promoting informed debate [7]. The experiences in IoT seem to show that beyond the energy sector, particularly in the arena of consumer electronics, there are still similar lessons to be learnt in bridging this security divide.

16.3.2 Device Challenges

The nature of some IoT devices can itself create a security design challenge. A very small device and/or one that is operating away from a source of electrical power may only have limited processing capabilities, and so standard solutions such as public key encryption may be difficult. A very large deployed base can also present cost challenges where even a few pence on the cost could impact a business model. These 'constrained devices' are becoming sufficiently important that they have become the subject of an Internet Engineering Task Force (IETF) Request for Comment looking at use cases, deployment and management options that include security constraints—RFC7547 [8].

16.3.3 Architectural Challenges

The architectural challenge for security comes about from the assumed conceptual model adopted by most information security professionals that is arguably 'baked into' the approach and strategies adopted for managing information security. This concept is based on the assumption of a safe company network that is based inside of a corporate perimeter. Even with the significant growth of mobile apps and the cloud, the focus and expertise of corporate IT security has remained largely on concerns that are inside the company firewall. But this new IoT environment of 'things' resides almost entirely outside of the corporate network, relying primarily on the external internet and local connectivity technologies such as Bluetooth or mesh networks. Security capabilities such as asset management, monitoring by network intrusion detection or having filters and trigger points at network gateways do not exist, or are difficult to deploy, outside of the corporate network; thus leaving the security practitioner without their usual toolsets.

16.3.4 Unclear Responsibility for Maintaining Security

The earlier examples of systems not being kept patched and up to date raises the fundamental question of who should be responsible for managing the security of deployed IoT. Where systems are inside of the corporate network or where they are part of company managed distributed systems (such as bank cash dispensers or oil field sensors), it is usually clear that the company is responsible for both the definition and maintenance of security. Where systems and devices are purchased individually by an end customer, particularly by a consumer, then this responsibility is less clear.

In the history of personal computer ownership, tasks like patching were originally manually applied, but in 2004, Microsoft introduced the concept of security status and update management with the release of Windows XP Service Pack 2. But patching responsibility for home systems was still in the hands of the end users who were expected to check the status and permit updates to happen. The following decade was still marked by poor patching of home systems, some because of the manual update processes still required by applications such as Flash and Acrobat, and some through end-users still not enabling or initiating Windows updates. This was resolved in the 2015 release of Windows 10 which does updating automatically and without expecting user interaction.

Unfortunately, current strategies for some IoT systems—and even proposed by some security solution vendors—repeat the pitfalls of the past, such as proposing that end-users take responsibility for security configuration and patching. As outlined above, the track record of end-user system security management does not inspire confidence, but even if it were better, end-users can hardly be expected to support the scale and complexity of IoT system deployment when they may not even be aware that a new appliance is in fact a fully networked computer. Where there are multiple vendors in user-selected open systems approach (as illustrated in Fig. 16.2), things can become very confused.

If we determine that suppliers will need to take on the responsibility for systems security, then IoT can present us with problems of precisely which organisation in the supply chain should take on the different roles. Those deploying IoT may typically have the devices designed and supplied by OEM device manufacturers, and so security changes need to flow through the supply chain.

In the case of the home environment, some integrated IoT systems are managing their own security status, but some are not, leaving the current end-user a degree of technical work to find out security and patch status. Although, as argued above, this is not a sustainable position, and we may see regulation for critical applications that mandates some form of security assurance and labelling, analogous to what we see for equipment safety with CE (‘Conformité Europèenne’) Marking or the UK safety ‘Kite Mark’. For security such a mark would need to recognise both secure design and also secure maintenance and management.

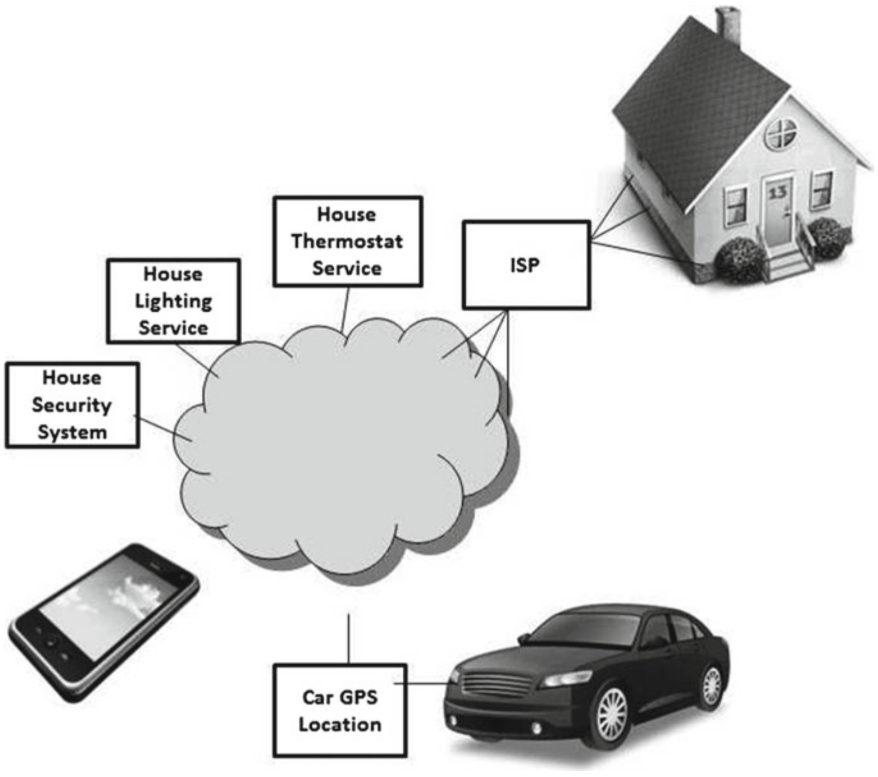


Fig. 16.2 For an open systems approach, who is responsible for security?

16.4 Why We Really Do Need to Get Security Right

A review of the need for improved security in the deployment of IoT causes us to re-visit some fundamental good ideas in security which have had relatively poor or slow take-up in the IT environment.

The key concept here is the need for robust systems created with ‘security by design’ where security functionality is designed in and development pitfalls introducing security weaknesses from coding errors are avoided. There may even be a resurgence in interest in the independent security testing of products, in a similar way to safety (see above).

An obvious challenge to this view is that if we have seen poor take-up of good security practices in the past, then what is different about IoT that will change the outcomes?

It is the opinion of the author that the sheer scale of the proposed number of IoT distributed systems relative to past IT deployments will create a situation where we would not be able to catch up and retrofit security. The type of functions that some IoT systems perform also introduce the possibility of security breaches resulting in physical damage or harm to individuals.

In June 2015, a robot in a German automotive manufacturing plant regrettably crushed and killed a worker [9] who was maintaining it whilst the safety software was disabled. The robot was not capable of acting autonomously (making a learning decision), and the death was not caused by a security breach, but the IoT security analogy—of a security breach having the potential to physically harm—is clear to see.

Risks like these have to be taken seriously and are likely to bring in regulatory intervention if incidents start to happen.

16.5 Addressing the New IoT Security Challenges

Having made the case in the earlier part of this chapter, the following sections aim to draw together all of the different IoT considerations and look at the fundamental conceptual challenges. The main dimensions of this new security situation are summarised in Fig. 16.3.

16.5.1 The Need for an IoT Conceptual Model

Arguably, most of the previous examples of security concerns are not unique to the IoT as they are also faced in mainstream IT. However, as mentioned earlier, in order to adequately address IoT security, we need to ensure that we have correctly defined both the scope and the likely use cases.

It is often tempting for an organisation to define its IoT security requirements purely by the first application that emerges—be it extending industrial control to the internet or connecting a mobile phone to a new wearable technology. However, the initial set of IoT applications is likely to expand significantly over time.

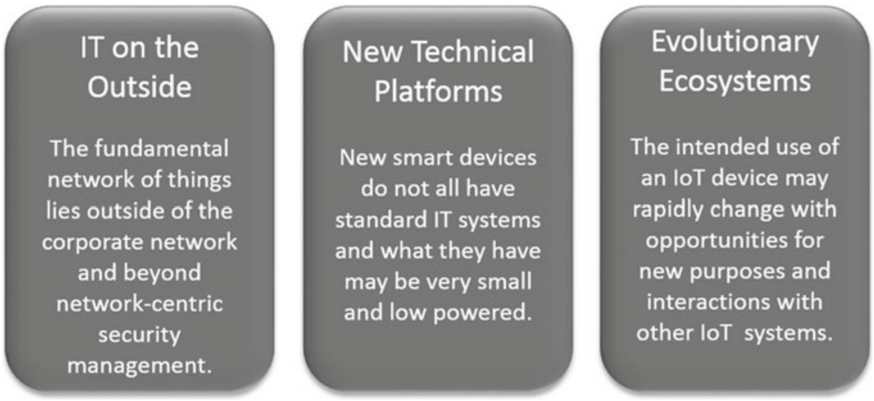

IT on the Outside
The fundamental network of things lies outside of the corporate network and beyond network-centric security management.
New Technical Platforms
New smart devices do not all have standard IT systems and what they have may be very small and low powered.
Evolutionary Ecosystems
The intended use of an IoT device may rapidly change with opportunities for new purposes and interactions with other IoT systems.

Fig. 16.3 The IoT security challenge

Similarly, the initial IoT security focus is most often on the ‘micro element’ of the device even though we know that most devices cannot securely operate without the ‘macro elements’ of applications, data stores and services which bring trust and security to this distributed world of ‘things’. The macro elements must be in place, because a device on its own does not know if it can trust messages from other components or services unless it has them authenticated by an authority that it can recognise. A trusted messaging channel/cloud service(s) that allows one device to accept commands or release data based on the state of another is fundamental to being secure.

Whilst some macro elements can work effectively in closed systems where devices maintain their unique identity and security status through a dedicated management system, they become more powerful when one device is used for more than one purpose, and thus, a device needs a unique identity and common authenticator that multiple applications can recognise. The conceptual model shown in Fig. 16.4 may be helpful in realising that there are broader security challenges beyond the device alone.

This model highlights the following managerial and technical elements and challenges:

Statement of company values The stance that the organisation wishes to take in terms of privacy, security and the assurance that its standards are being met.

Independent verification Independent assurance and/or testing that IoT security has been and is being effectively addressed in the development, deployment and operation of a new system.

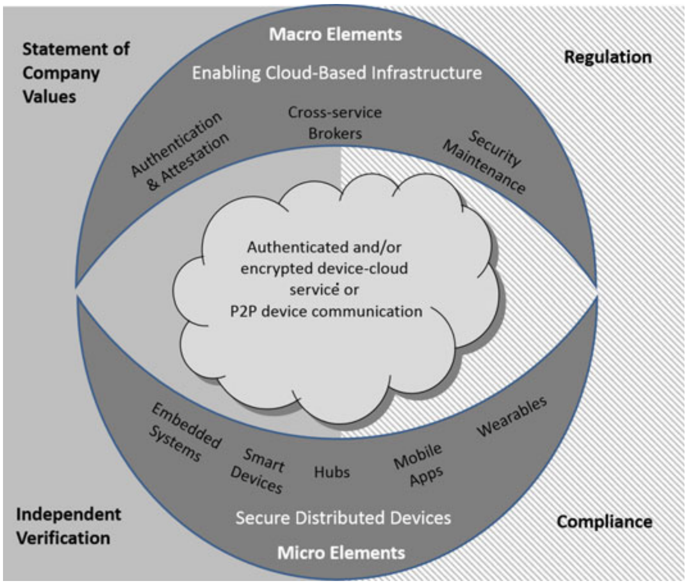

Statement of Company Values
Macro Elements
Enabling Cloud-Based Infrastructure
Regulation
Authentication & Attestation
Cross-service Brokers
Security Maintenance
Authenticated and/or encrypted device-cloud service or P2P device communication
Embedded Systems
Smart Devices
Hubs
Mobile Apps
Wearables
Secure Distributed Devices
Micro Elements
Independent Verification
Compliance

Fig. 16.4 IoT conceptual model

Regulation and compliance Covering both general regulation such as protection of personal data and specific regulation such as safety or medical systems integrity as required by a particular regulator.

Authentication and attestation Security capabilities that can give confidence in the source (identity) of an IoT component and the validity of the messages it is receiving or transmitting.

Cross-service brokers Services that provide software logic that makes decisions or triggers actions, such as IoT outputs based on IoT inputs.

Security maintenance The processes and technical management actions that allow security weaknesses in software or its configuration to be corrected.

Embedded systems IoT technologies that are built into existing systems to enable them to communicate over the internet and provide data or respond to commands.

Smart devices Devices capable of data processing and designed from the outset with internet connectivity.

Hubs Smart devices such as smart phones that provide an interface between less sophisticated IoT components and services on the internet.

Mobile apps Software designed to run on smart mobile devices. Apps are often the human interface used to control IoT systems and applications.

Wearables Smart devices that are worn to provide additional sensors or capabilities (for example glasses as video displays and watches as heart rate monitors). These may connect directly to the internet or via a smart phone hub. They will also increasingly connect to each other.

16.6 Formulating an IoT Security Strategy

IoT security is sufficiently important that the actions and decisions that companies make about it will often shape their reputations and future business opportunities, as further discussed below:

16.6.1 Reputation and Stance

IoT has the potential to benefit the supplier, the customer, or both. But we cannot assume that these benefits will be automatically available. Particularly when IoT services are used by specific individuals, explicit permission is generally needed for data collection and use. Even the collection of non-personal data often needs at least tacit agreement from a broader set of stakeholders, possibly including regulators.

Getting these issues wrong could have commercial implications as customers may reject products or services seen as untrustworthy. Mistakes can even result in regulatory intervention. There are two main scenarios where permissions are important:

1. A company collects and uses customer information to help in managing its own business (for example proactive maintenance, cross selling or on-selling of data).
2. A company collects information to provide additional/improved functionality and services of potential benefit to its customers.

Transparency of actions and clear statements on data use and security are therefore fundamental, even when there are no underpinning laws and regulations.

16.6.2 Fair Use or Abuse of Data

Before we examine the obvious concerns of personal privacy, it is useful to look at the potential value and risks in any data being collected. The very fact that data collection has benefit to someone raises the possibility that the same data could be of interest to someone else. If we take the example of agricultural IoT applications where crop environment and health can be measured to help the farmer, the same data could arguably help predict production levels and be of great interest to commodity traders. Uncontrolled release or access to such data could therefore have market implications that may even disadvantage the farmer and potentially breach market regulations. Some suppliers of agricultural IoT data services have become aware of these sensitivities and have created a code of conduct stating that data will not be shared with anyone other than the direct customer who has deployed the sensors. In the future, the need for independent assurance of such statements may be expected.

16.6.3 Privacy

Of course, collecting personal data is not new in a world of loyalty cards and web usage tracking, but the pervasiveness, volume of data, and physical proximity of IoT to individuals can make the subject very emotive. For example, Ford Motor Corporation was widely criticised in January 2014 after an executive publicly stated that Ford could (in theory) monitor speeding infringements made by its customers [10]. In November 2014, the relevant U.S. automobile manufacturing associations issued a letter and code of practice to the U.S. Federal Trade Commission on how in-car technologies would be used and protected [11].

Similarly, at least two TV manufacturers have been taken to task for using cameras and microphones to monitor more than their customers expected [12]. We tend to think that companies have a legal obligation to declare or explicitly ask permission in order to collect and use such data, but, for example, when the information is only used in aggregate, covert data collection can (at least arguably) be outside of current regulation. In such cases, each company deploying or supplying IoT systems must set its own standards and take its own stance.

Professional IoT security could help in these areas by demonstrating that Privacy attestations are being honoured, and in giving confidence that there is substance behind any claims that data is protected. But first and foremost, a company deploying IoT capabilities needs to decide what its stance will be on data collection, use and sharing, as well as the security performance standards that it will meet.

16.6.4 Liability

To date, with the exception of data protection/data privacy incidents, there have been few examples of companies being held liable for the failure of their IT systems. Whether this will change as more physical systems become part of the digital world is currently unclear, but there is definitely growing interest from bodies such as the US Federal Trade Commission and the European Commission. It does not take much imagination to see that companies will need to demonstrate that they are following best security practices in the use of IoT as they emerge. It can be expected that the growing cyber insurance market will also push for greater clarity.

16.6.5 Privacy and Society

However, beyond the views of a single organisation, there is also a broader concern of the possibility of data aggregation from multiple services—such as we are already starting to see with security or traffic monitoring in smarter cities. In these cases, the risk is not just that the organisation or government body doing the collection and analysis of data may abuse the information and go beyond expected use, but also that such data collections could be leaked, stolen or accessed by others such as criminals. The risks of mass data aggregation are not as well researched as they could be, but it is clear that even anonymised data may be attributed by combining with other contextual data. A very personal profile could be developed by Big Data combinations of location, habits, health, purchases, contacts, family relationships etc. When everything we do has a digital component either directly or through the systems, CCTV cameras, microphones or tracking systems owned by others around us, the correlation possibilities are almost endless. At the moment, there seems to be no clear solution to the problem of widespread loss of privacy unless we can come up with some way of creating ‘privacy firewalls’.

16.6.6 Selection of Solutions and Standards

Many companies will choose not to develop their own IoT solutions. Instead, they will rely on the suppliers of OEM devices, software development toolkits and even cloud services vendors supporting IoT deployment. Thus, vendor selection is now a key IoT security component because of the wide range of standards and standards groups that are currently active. Even the service companies themselves often struggle to determine the standards and ecosystems they wish to support.

Standards are particularly important for interoperability, as the ability to reuse macro security services can only come from standardisation and agreed common approaches. Unfortunately, there are too many standards initiatives at the moment as the major vendors jostle for position. Often, it is not clear which IoT consortia to follow and what standards to adopt.

At the time of writing, examples of industry initiatives include the AllSeen Alliance (including Qualcomm, Cisco, Microsoft and many consumer product vendors), OIC—Open Interconnect Consortium (Dell, Samsung, Cisco and others), Thread Group (including NEST and ARM), HyperCat (including ARM, BT and Rolls-Royce), HomeKit (Apple) and the Industrial Internet Consortium (including GE and AT&T). There are also specific initiatives in healthcare, energy (smart grids and smart meters) and automotive (such as MirrorLink). More broadly, some of the thinking about security outside of the corporate network that originated in the Jericho Forum of the Open Group is being continued by the Global Identity Foundation.

Most of these groups are still in their early stages and they should be challenged on the design of their security capabilities and questioned regarding secure development and ongoing security management. The current situation remains confusing, and these issues need to be simplified if the IoT is to fulfil its potential.

More general security guidance and standards are also in the process of development, including Security Guidance for Early Adopters of the Internet of Things (IoT), produced by the Cloud Security Alliance [13].

16.6.7 Gateways and Aggregators

The use of gateways and aggregators is a natural solution in a closed system of relatively simple devices that need to communicate more robustly [14]. A gateway approach could perhaps also be the answer to the apparent open-system anarchy of the unmanaged home environment of consumer devices. E.g. does a fridge have a legitimate reason to communicate with a TV, especially with what looks like a denial of service attack? One can readily see the benefit of some independent security assurance hub that could assess and manage the security status of IoT devices linked to that hub. At the time of writing, there are no solutions available, although some have expressed hope that new hubs such as Google on-hub or Amazon's Echo might be developed to provide these services. Perhaps current Internet Service Providers will see an opportunity to increase the security capability of their network routers and provide a security service?

16.7 Development and Operation

The examples of IoT security lapses described earlier in this paper show that firms need to consider what security functions an IoT system can perform, how security and integrity will be maintained and how well systems are developed to avoid security vulnerabilities. The author recommends focusing on the following development and operating principles:

Take an end to end view of risk Risk will need to be assessed and modelled to build security across the entire data flow, including devices, applications, storage, brokers and apps. There should be no 'weakest link' that is not identified and understood.

Secure development The components of IoT should be developed using robust development methods to reduce vulnerabilities, and the security of each component then independently tested.

Maintain integrity The systems provider or someone else expressly identified in the supply chain should accept responsibility for on-going security management; this task should not be delegated to the end customer if they are consumers.

Preserve 'agency' and control Systems should be deployed so that they only accept instructions both with the explicit consent of the customer and through controlled channels that have been authorised by the system vendor.

Build in resilience System components should be designed to operate in an environment of hostile devices, and these components should be capable of detection of compromise and restoration to a trusted state following any compromise.

Maintain future trust Security should be designed with extensibility so that the implementation or subsequent upgrade can eventually support the most sensitive anticipated usage.

16.8 Assurance

In addition to a company complying with its own security principles, there should be a level of independent assurance verification so that the customers of the company can more confidently trust the integrity of the system. Within complex ecosystems, such assurance becomes even more important as each ecosystem partner needs to demonstrate that it meets the expected security standards for trusted information exchange. Eventually, this type of interoperability requirement and certification may well become part of the cyber insurance industry.

16.9 Evolution

Many of the most exciting IoT benefits come from automatic machine-to-machine communication—from one type of 'thing' talking to another, such as a home thermostat that turns on the heating system when your phone's GPS knows you are heading home. Many such automatic intervention services will be possible as more IoT and communications capabilities are deployed.

Many risks come from a situation where the trustworthiness of one ‘thing’ is not consistent with the security needs of another. For example, there is clearly a close connection between fitness and health, and a fitness sensor monitoring heart rate can provide valuable data to a medical application. But what happens if the fitness device vendor has not adequately protected its own data or device, or if it has a different privacy policy, and sells data to insurance or marketing companies? This juxtaposition of critical/regulated applications with those of hobbies/leisure is just one illustration. Security strategies must take into account the likely evolution of the purposes to which an IoT system will be put, and the future stakeholder environment it will be part of. These issues are especially important in regard to regulation.

But the regulation of IoT devices is still embryonic. Of course, general regulations such as data protection will apply when personal data is being captured and manipulated, but specific sector regulators are still considering the various IoT implications. IoT vendors must consider the implications of future imposed regulatory standards on distributed systems and even the impact of a change of purpose of an IoT system, as illustrated in the following diagram (Fig. 16.5):

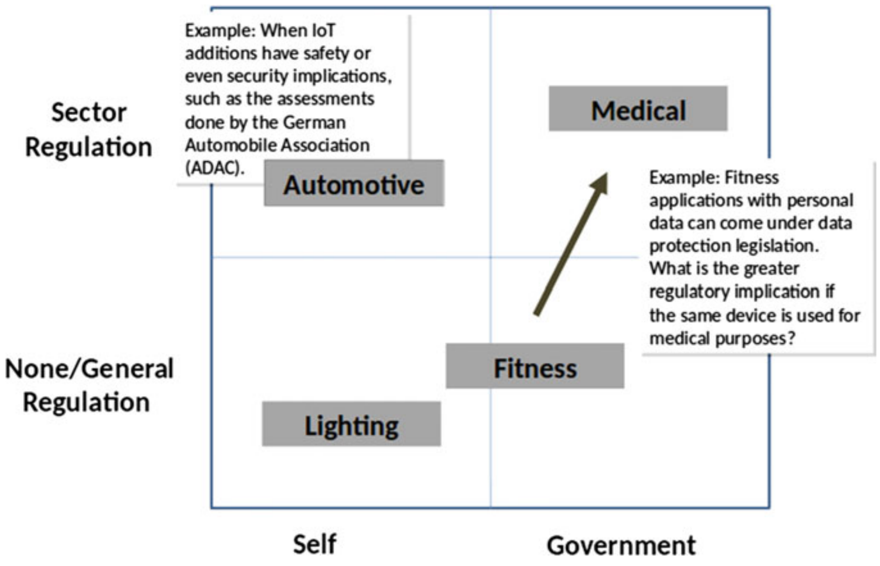

Example: When IoT additions have safety or even security implications, such as the assessments done by the German Automobile Association (ADAC).
Sector Regulation
Medical
Automotive
Example: Fitness applications with personal data can come under data protection legislation. What is the greater regulatory implication if the same device is used for medical purposes?
None/General Regulation
Fitness
Lighting
Self
Government

Fig. 16.5 IoT regulatory positioning

16.9.1 *Open Versus Closed System*

Closed systems which are dedicated to one vendor and a limited set of applications may have security and risk models which are easier to manage, but commercial opportunities may be missed if additional value cannot be added by other uses for the deployed systems. If a decision is made to adopt a more open-system approach, then the choice of standards and consortium membership becomes even more important. As mentioned earlier, open systems of multiple vendors raise the question of who is responsible for end-to-end security.

16.10 A Framework for Defining and Measuring IoT Security

Because of the speed of evolution of the IoT companies would be well advised to start working on their IoT security strategy now, before they become overwhelmed by the pace of digital business and the many tactical decisions required. The following model draws on information from earlier in this chapter and has been developed to help organisations shape their IoT security strategy and assess IoT projects as they arise. The model has four main dimensions: strategy, development, operation and evolution—each emphasising three high-level questions, as shown below (Fig. 16.6).

To examine these questions and some of the underlying thinking in more detail:

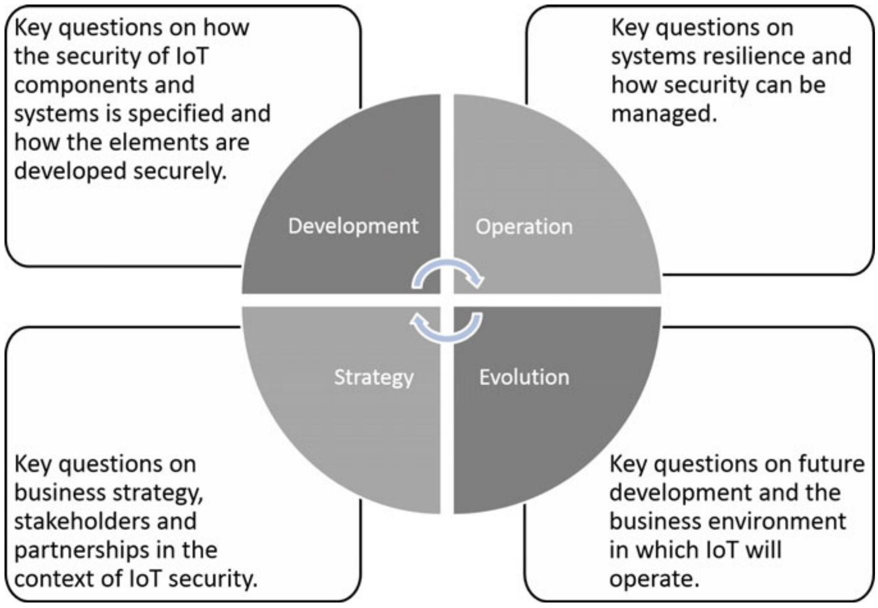

Key questions on how the security of IoT components and systems is specified and how the elements are developed securely.
Key questions on systems resilience and how security can be managed.
Development
Operation
Strategy
Evolution
Key questions on business strategy, stakeholders and partnerships in the context of IoT security.
Key questions on future development and the business environment in which IoT will operate.

Fig. 16.6 A framework for defining and measuring IoT security

16.10.1 Strategy Questions

1. *Has the company decided on its IoT leadership and goals in the industry?*

Decisions that will need to be taken will include the company stance on privacy and how data will be used, and may even include commitments to independent assurance that obligations are being followed. Some may decide to take a particular high-ground and/or work with regulators in devising standards.

In quite a number of cases, there may even by a requirement to establish an ethics committee to oversee privacy implications or even the implications of autonomous actions taken by IoT devices, agents and brokers. Even today many Internet/mobile device users find themselves working through software which makes decisions and filters data on their behalf and there has been little work on how assurances of ‘fair play’ or ‘acting in the users best interests’ can either be defined or communicated.

2. *What standards and consortia/ecosystems should be followed?*

Some standards will gain better deployment and take-up than others, and in terms of security, some will build-in better capabilities than others. standards are also likely to be key in how the different responsibilities of end-user, service provider, developer or component manufacturer, are defined and articulated to ensure we have no gaps in security capability.

3. *What is the risk assessment covering current and future deployment?*

As risk assessments are founded upon the business impact and implications of a security breach for particular data and business purpose it can be particularly difficult to assess risks for future purposes and use cases. For IoT, this prediction may be a key requirement and more work on open-ended or future-proof security modelling is needed. Some think that this will be as much about the ability to respond to and recover from security breaches as it is about relying on long-term protection mechanisms.

16.10.2 Development Questions

4. *Is a secure development life-cycle used for all components?*

A robust development lifecycle with developers who are well trained in threat modelling (e.g. methods such as STRIDE [15]) to eliminate security vulnerabilities will help ensure a robust and resilient code base for distributed and centralised IoT. Current development practices are well below the necessary standard for resilient systems, and we may need actions such as a significant refresh of the common open source code libraries being used in these developments.

5. *Are the IoT components independently tested?*

Testing in accordance with security criteria has not been well adopted in the IT community but is normal in the context of safety testing in the engineering community. The establishment of more test labs capable of testing the security of IoT modules would be a key capability. Currently, there have been some services to test industrial control systems but take-up is relatively low.

6. *Does the IoT system have the necessary security designed-in?*

A security review that looks for adequate security functionality (e.g. encryption, authentication, access control etc.) should be performed for all IoT systems and assessed against security requirements. This is a key topic and is therefore worthy of some further exploration:

Why not just use normal Internet security?—As properly managed Internet security has proved to be reasonably effective against a range of security threats, it could be argued that IoT devices just need to be given an IPv6 address and properly configured. However, things are not so straightforward, not only because many devices are being set up with IPV4 and are running behind firewalls with Network Address Translation, but also because of challenges in security management and device performance:

- Security management is challenging because most IoT systems will not have the level of user interaction we see with PCs and mobiles and so prompts for local user activity such as applying software updates cannot be relied upon. The large volume (billions) of devices also stretches manageability even in closed systems where the service provider is clearly on the hook to manage their own systems. Even telecommunications providers have never seen such volumes before.
- Manageability becomes even more difficult with open ecosystems where each individual component may need security maintenance from its own component supplier. Some service capable of supporting security management and assurance across vendors (as we see in enterprise IT) may be required, but we do not yet have the necessary core standards nor stability in what to expect in IoT platforms.
- Low powered, low cost and/or low power consumption applications also reduce the capabilities available in some IoT devices. For example, in one smart meter application, the meter is required to have 15–20 years of battery life, and it has been calculated that a single software update could consume many months of that life. Simple devices also may not have IP protocol stacks and instead have proprietary protocols and gateways.

What about security modules?—The standard solution to achieve distributed security is generally to have security modules embedded in such devices. Examples of this include smart cards, Secure Execution Environments (SEs), mobile SIMs and e-passports. These modules provide an attack-resistant trust anchor to allow trusted device identification and personalisation and can protect stored data, cryptographic algorithms and keys/credentials. This is a tried and trusted approach but really has only been proven in closed ecosystems with a relatively small set of issuers (e.g. banks or telecommunications providers) buying and controlling the devices within agreed industry standards and frameworks. So what about the IoT?

The IoT currently has some closed applications, such as electricity and gas smart meters, and these are using security modules. There is also some piggyback activity on standards such as GSM where SIM cards from mobile network connected IoT devices can also provide a kernel for their general security [16]. However, in the main, most IoT devices are not being deployed using security modules in their design, even if the processor chipset being used is capable of providing this capability. The Trusted Computing Group [17] are working on this but we have seen a similar resistance to using available trusted capabilities in standard PC deployments where there are not as many constraints.

16.10.3 Operations Questions

7. *Can the systems have their security easily (automatically) managed and updated?*

Our experience in security management of current IT and OT/SCADA systems tells us that we should expect that robust security management and patch management is performed to reduce the potential of vulnerabilities being exploited. However, the IoT has large numbers of distributed and embedded devices that again present some challenges:

Initialisation and personalisation needs to be done without knowing the eventual owner/user and probably even the application. The device can perhaps be personalised with a unique device ID, and also with security keys that could be subsequently used to do remote personalisation and security management. In some applications, personalisation involves linking the user with the device through a web portal. Where keys are used, there may be limitations on having to use symmetric management keys because a low powered device cannot support the asymmetric public key processing.

Personalisation matters because maintenance and update requires a decision to be taken on how the device can trust the update being provided and what involvement the owner/operator of the IoT device should have in agreeing to the nature and timing of any update. For example, if a car was to receive an Over the Air update it would perhaps not be wise to apply it whilst the car is in motion. False software updates are also a well-known attack vector, and so integrity and authenticity of genuine updates will need to be assured.

8. *Can the system resist the risks on being on hostile network?*

As they will mainly reside outside of corporate networks, we need to assume that the IoT devices will be connected to the raw Internet or be sited next to other devices that have themselves been compromised. This suggests that the device may need to be capable of being told what an attack looks like and of communicating security events when it sees them. In an open IoT ecosystem, this also suggests that devices will need to be capable of communicating their trust status and identity to other devices so as to be able to interoperate on a hostile network. The protocols to achieve this are still in need of development with the initial work on IoT security primitives by NIST being worthy of mention [18].

9. *Can compromise be detected and is there a fall-back for recovery?*

And once a device is compromised, we should look to see if we can restore it, ideally remotely because physical access to the device or visiting a number of devices may be difficult after deployment.

16.10.4 Evolution Questions

10. *Have we created the de facto standard or are there better industry solutions emerging?*

Selecting the winning standard or consortium membership may be elusive, and it may be necessary to revisit these decisions in time. In fact, we have already had a limited illustration in the case of one home IoT system which was made obsolete when the supplier not only stopped software maintenance, but also closed down the cloud service the devices depended on to operate [19].

11. *Are use cases and interactions evolving beyond those first envisaged?*

As covered in question (3), one of the most difficult tasks is to design a future-proof security system that covers future use cases and future threats. We should expect to revisit this periodically, but it is still a design stretch to have a device completely future-proof against new risks and threats. Existing industrial control/SCADA systems are addressing the challenge of new risks through architectural design, so we can imagine some scenarios where new security challenges may be able to be addressed at a hub rather than at every remote device. This still remains an area of design challenge and presents an opportunity for future research.

12. *Are our leadership goals aligned with emerging regulations?*

We can expect regulations to change, driven by public opinion and events, and companies may need to adjust their position accordingly. The industry sectors with heavy regulation (such as pharmaceuticals or financial services) should expect early action by their regulators. However, more broad regulation should be expected to happen across all sectors.

There are numerous other questions that could be asked by organisations embarking in IoT deployment, but these particular ones have been selected as key, and across all of the dimensions that should be considered.

16.11 Conclusions

The emergence of the Internet of Things represents an even bigger step change to the corporate information security environment than we saw with the growth of cloud computing. The security challenge that this presents will not be solved by purely focusing on the technology. IT leaders should therefore be engaging as early as possible with interested business colleagues, and devise an approach that articulates a clear company IoT security strategy, positions the company within the IoT ecosystem and puts in place the benchmarks and processes needed to keep IoT projects and initiatives on track. In the view of the author, the time to begin this work is now because the fundamentals will take time to develop.

The IoT technology providers should step up and respond to the requirements of their commercial customers and agree standards and approaches that allow security to be built into systems from the start. In some instances, this will require some step-out thinking and research not just into the technical security challenges, but also how approaches to concerns such as privacy or behaviours of systems on behalf of their users can be articulated, attained and assured.

Acknowledgements The author wishes to thank Professor Keith Mayes of Royal Holloway, University of London, as well as David Moschella and Adrian Seccombe of the Leading Edge Forum for their help in developing some of these ideas.

References

1. Cisco Systems, Inc, Available: http://ioeassessment.cisco.com/learn/ioe-faq, 28 September 2015, [Online], [Accessed 28 September 2015].
2. ITU-T, Available: http://www.itu.int/rec/T-REC-Y.2060-201206-I, 15 June 2012, [Online], [Accessed 28 September 2015].
3. Gartner, Available: http://www.gartner.com/newsroom/id/2905717, [Online].
4. Cisco Internet Business Solutions Group (IBSG), D Evans, "The Internet of Things—How the Next Evolution of the Internet Is Changing Everything", Cisco IBSG, 2011.
5. C. Miller and C. Valadek, "Remote Exploitation of an Unaltered Passenger Vehicle", 10 8 2015, Available: www.illmatics.com/Remote%20Car%20Hacking.pdf, [Online], [Accessed 17 October 2015].
6. OWASP, "OWASP Internet of Things Project", 2014, [Online], Available: https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project, [Accessed 30 4 2016].
7. S. Peisert, J. Margulies, E. Byres, P. Dorey, D. Peterson and Z. Tudor, "Control Systems Security from the Front Lines", Security & Privacy, vol. 12, no. 6, pp. 55–58, 2014.
8. M. Ersue, D. Romascanu and J. Schoenwaelder, "Management of Networks with Constrained Devices:", May 2015, [Online], Available: https://tools.ietf.org/html/rfc7547, [Accessed 29 September 2015].
9. E. Dockterman, "Robot Kills Man at Volkswagen Plant," Time Life, 1 July 2015, [Online], Available: http://time.com/3944181/robot-kills-man-volkswagen-plant/, [Accessed 17 October 2015].
10. J. Edwards, "Ford Exec: 'We Know Everyone Who Breaks The Law' Thanks To Our GPS In Your Car", Business Insider, 8 January 2014.
11. Auto Alliance & Global Automakers, "Consumer Privacy Protection Principles for Vehicle Technologies and Services", 12 November 2014, [Online], Available: https://www.globalautomakers.org/system/files/document/attachments/Global%27s%20and%20the%20Alliance%27s%20FTC%20Letter%20Committment%20and%20Privacy%20Principles%20%282%29.pdf, [Accessed 29 September 2015].
12. T. Danielson, "Samsung explains why its SmartTV records private conversations (+video)", The Christian Science Monitor, 9 February 2015.
13. Cloud Security Alliance, "Security Guidance for Early Adopters of the Internet of Things (IoT)", April 2015, [Online], Available: https://downloads.cloudsecurityalliance.org/whitepapers/Security_Guidance_for_Early_Adopters_of_the_Internet_of_Things.pdf, [Accessed 29 September 2015].
14. F. daCosta and B. Henderson, Rethinking the Internet of Things: A Scalable Approach to Connecting Everything, Apress, 2014.
15. F. Swiderski and W. Snyder, Threat Modelling, Redmond: Microsoft Professional, 2004.
16. GSM Association, "GSMA IOT Guidelines Complete Document Set", February 2016, [Online], Available: http://www.gsma.com/connectedliving/gsma-iot-security-guidelines-complete-document-set/.

17. Trusted Computing Group, "Guidance for Security IoT Using TCG Technology", 14 9 2015, [Online], Available: http://www.trustedcomputinggroup.org/wp-content/uploads/TCG_Guidance_for_Securing_IoT_1_0r21-1.pdf, [Accessed 30 4 2016].
18. J. Voss, "Primitives and Elements of Internet of Things (IoT) Trustworthiness", February 2016, [Online], Available: http://csrc.nist.gov/publications/drafts/nistir-8063/nistir_8063_draft.pdf, [Accessed 30 4 2016].
19. K. Finley, "Nest Hub Shutdown", Wired.com, 4 5 2016, [Online], Available: http://www.wired.com/2016/04/nests-hub-shutdown-proves-youre-crazy-buy-internet-things/, [Accessed 30 4 2016].
20. Hewlett Packard Enterprise, "Internet of Things Research Study 2015", [Online], Available: http://www8.hp.com/h20195/V2/GetPDF.aspx/4AA5-4759ENW.pdf [Accessed 30 4 2016]

Chapter 17
MULTOS and MULTOS Application Development

Chris Torr and Keith Mayes

Abstract This chapter discusses the concepts behind the MULTOS smart card operating system and introduces the development environment and tools. It also briefly discusses possible future uses for MULTOS outside of smart cards.

Keywords MULTOS · Multi-application · Operating system · High security · Remote provisioning

17.1 Introduction

MULTOS is a high security, multi-application operating system designed from the outset for smart card products. In the first instance, this was for the electronic purse, Mondex, where the card stores digital cash. Although Mondex was not a commercial success at the time, the operating system on which it was built has gone on to be used extensively in many different fields. Predominantly MULTOS is used today for banking applications (EMV) but also has a strong presence in ID (including passports). It is used for many other applications too such as loyalty, driving licences, social and health entitlement.

MULTOS is an open standard. That is to say, that the standard is governed by the MULTOS Consortium which consists of some thirty global companies who provide MULTOS related products and services. It is the role of the consortium to develop and evolve the operating system to meet the demands of new products and markets. The consortium is managed by a secretariat company, MAOSCO Ltd, whose role is to run the consortium, promote MULTOS, supply developer tools and ensure compliance of products to the MULTOS specifications.

C. Torr (✉)
MAOSCO Ltd, London, UK
e-mail: chris.torr@multos.com

K. Mayes
Director of the Information Security Group, Head of the School of Mathematics and Information Security, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK
e-mail: keith.mayes@rhul.ac.uk

K. Mayes and K. Markantonakis (eds.), *Smart Cards, Tokens, Security and Applications*, DOI 10.1007/978-3-319-50500-8_17

17.2 Definitions

Note: terminology used in this chapter is shown in Table 17.1.

17.3 What Is MULTOS?

As already stated, MULTOS is a multi-application operating system. The applications can already exist in the device or be loaded/deleted dynamically. Application development is usually done in ‘C’ or assembler code (MEL). the MULTOS architecture is shown in Fig. 17.1.

MULTOS applications are written to run on the MULTOS Application Abstract Machine (AAM). This is a virtual machine coded to run on the native execution environment of the underlying secure micro-controller chip. It has a rich function library available supporting the common operations needed by a smart card, such as cryptography, PIN handling and memory management. Applications are therefore portable between devices from different implementers. The type approval scheme, run by MAOSCO, enforces the interoperability of MULTOS devices supplied by different vendors.

The virtual machine contains a set of fixed instructions like you would see in a microprocessor (such as add, jump, branch, call, compare etc.) and a set of higher-level primitives.

Primitives are written in the native code of the microprocessor and are called using one of the AAM instructions. The architecture of MULTOS allows for new primitives to be easily added to the specifications. Implementers of MULTOS devices can also incorporate AAM code into the memory of the chip (traditionally Read Only Memory (ROM), but increasingly Flash memory) during its manufacture. These ‘codelets’ have several advantages such as faster personalisation time (no need to load code to each card, just data), protection of intellectual property, shared code libraries (codelets can be used to support common functions), cost reduction (cheaper to use ROM than EEPROM to store code) and making more EEPROM (or Flash) available for dynamically added applications. Memory layout is seen in Fig. 17.2.

In essence, the MULTOS O/S implements a run-time firewall between applications preventing applications from accessing the data of other applications. It is a true “multi-party secure” environment. There is no need to pre-check an application before loading it for safety of operation with existing applications because MULTOS ensures all memory operations remain in-bounds. In practice what happens is that the O/S creates a virtual address space for the application when the application is launched. This maps areas of the real, physical, memory to this virtual memory space. The bounds checking of the O/S ensures that all memory access remains within that virtual space.

Static memory is for data that is to be kept between sessions. It is allocated when the application is loaded. There is no concept of heap memory in MULTOS, although

Table 17.1 Table of definitions

Term	Definition
AAM	Application Abstract Machine. The virtual machine of the MULTOS operating system
AID	Application Identifier
ALU	Application Load Unit = Code + Data + DIR entry + FCI entry + Signature + KTU
APDU	Application Protocol Data Unit see [ISO7816-4]
ATM	Automated Teller Machine a cash dispensing machine
DES	Digital Encryption Standard a block cipher using symmetric keys
DIR entry	Application Identifier
EMV	Europay Mastercard VISA electronic payment standard
FCI entry	File Control Information a set of permissions associated with an application
FIPS	Federal Information Processing Standards a set of standards originating in the United States, published by NIST
GNU	A Unix-like computer operating system composed wholly of free software. GNU is a recursive acronym for “GNU’s Not Unix!”
HSM	Hardware Security Module a piece of electronics or computing equipment designed to securely hold secret data and perform encryption tasks
IDE	Integrated Development Environment a suite of software applications allowing a software engineer to write, compile and debug applications with a single graphical user interface
IFD	Interface Device—for example a smart card reader
ISO	International Standards Organisation
KMA	Key Management Authority. There is a “Global” KMA but it is also possible to have local, independent KMAs when countries/schemes require it
KTU	Key Transformation Unit contains keys and locations that the keys encrypt in the ALU. It is itself encrypted by the card public key
MAOSCO	Multi Application Operating System COmpany
MEL	MULTOS Executable Language. The “machine code” programming language of the MULTOS AAM
NFC	Near Field Communication short range radio communications conforming to various protocols such as MIFARE, ISO14443 where devices may be powered using radio frequency waves
NIST	National Institute of Standards and Technology United States standards body
PKI	Public Key Infrastructure methods for the trustworthy sharing of the public parts of asymmetric cryptographic keys
POS	Point of Sale
ROM	Read Only Memory—traditionally used in smart cards for storing the fixed code such as native applications or operating systems
RSA	Rivest Shamir Adleman public key encryption system
SDA	Static Data Authentication The original EMV mechanism for validating the data presented by an EMV smart card to a payment terminal
Unix	A family of multi-user, multi-tasking operating systems.

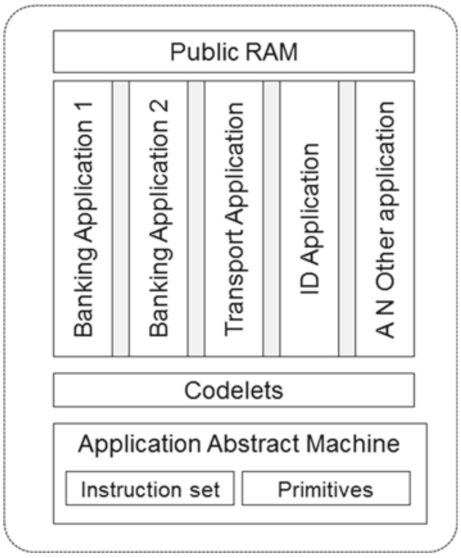


Fig. 17.1 MULTOS architecture

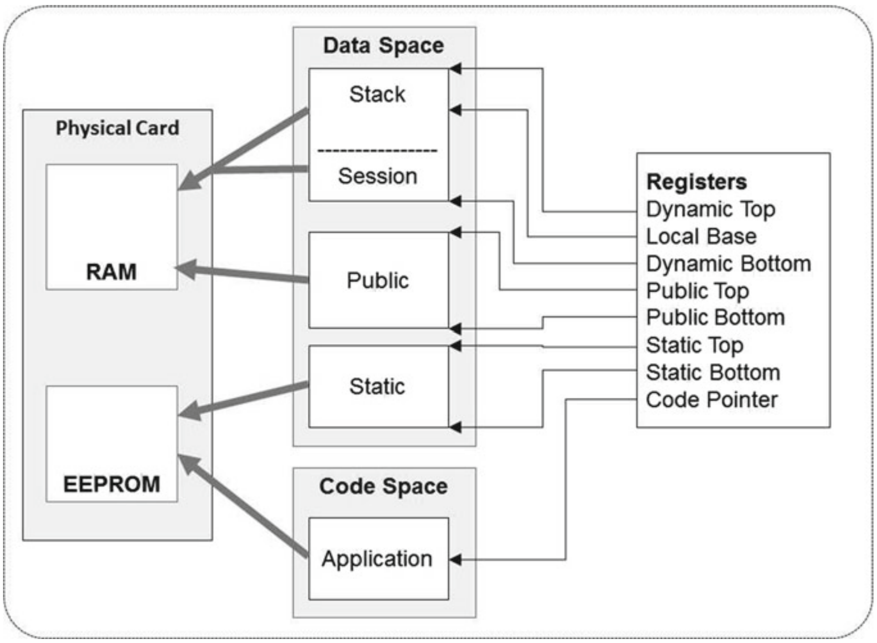


Fig. 17.2 Memory layout

you can if you wish write heap allocation routines that work within a predefined chunk of space. Because of the physical location of Static memory, it is not a good idea to access or update that memory constantly as it is relatively slow and, in the case of EEPROM, has a limited lifetime when it comes to changes. It is therefore much better then to perform calculations in Session memory, which as the name implies, is temporary and in RAM. Of course, all local variables are held on the stack which is also in RAM. RAM is usually quite limited though so a balance has to be found between optimised performance and having enough RAM for program execution.

Although a technical detail, it is worth noting that applications are able to access Static memory outside of the normal 64Kb virtual address space using a set of special primitives.

Applications can communicate with each other using the area of public memory used for communicating with the Interface Device (IFD). This inter-application communication is strictly controlled using a process known as Delegation where the state of the calling application (delegator) is preserved by the O/S whilst the called (delegatee) application executes. This is one way of implementing shared functionality and storage of information. As information is passed via the public RAM, it is of course important to ensure that the data is handled properly e.g. it may be worth encrypting it and the memory should always be overwritten when finished with.

17.4 Application Provisioning

MULTOS includes its own Secure, Trusted-Environment Provisioning mechanism (STEP). STEP is a comprehensive cryptographically-based mechanism that starts from the manufacture of each chip with the intention of being able to conclusively prove the authenticity and integrity of the chips and applications used whilst giving full control over card content to the issuer of the card. The MULTOS Issuance scheme is shown in Fig. 17.3.

There are two variants of this provisioning mechanism, the original "full" version based on asymmetric methods and the "light" step-one method based on symmetric methods. Both variants are very similar but the latter was introduced to run on smart card chips that do not have a crypto coprocessor (as used in EMV SDA cards). The rest of this chapter, however, deals with the full MULTOS method. Whether MULTOS, or step/one, once loaded, applications are developed and run identically. At the heart of STEP is the MULTOS Key Management Authority. As the name implies, this body is the source of all card keys and certificates used by the MULTOS scheme.

17.4.1 Manufacture

Security starts at the point of silicon manufacture. Unlike other smart card technologies, with MULTOS each chip is injected at manufacture time with a unique

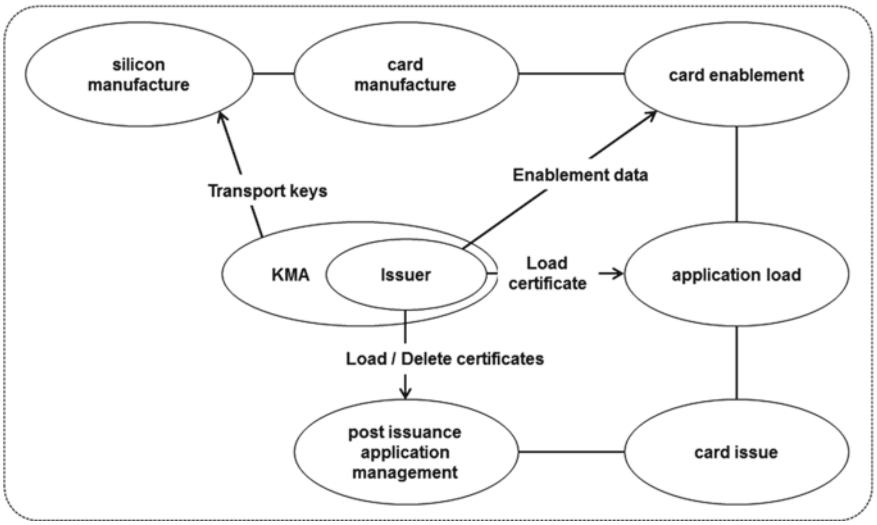


Fig. 17.3 MULTOS Issuance scheme

symmetric transport key derived from a key held at the KMA. It is also injected with a unique serial number (unique across all manufacturers). Therefore the wafers shipped from manufacture are highly secure with no potential for a class break. At this point the chips are truly generic stock items and can be used for any card issuer or use.

17.4.2 Activation

Once embedded into a card, the chips need to be activated (a step known as "enablement" in MULTOS parlance). This involves the card issuer (or their appointed agent such as a card vendor) requesting the enablement data from the KMA for the chips embedded in the cards. This is done by sending the list of serial numbers online (either in a batch or one-by-one via a web services interface) to the KMA. The KMA then sends the data back encrypted by the individual transport keys of the chips requested. This ensures that only genuine chips can be enabled. The data from the KMA is then loaded to the cards using a MULTOS specific APDU.

Following enablement, a chip is no longer a stock item as (a) it is assigned to the issuer (via an encoded ID), (b) it has its communications parameters set and (c) it has a unique, signed, key pair loaded to it (generated by the KMA). The public key certificate of the card is then all that is needed to be able to encrypt applications and data destined for that card. The card has an APDU command that allows you retrieve that key certificate (which you can validate with the KMA public key) so there is not even a need for the public key to be stored or distributed.

Having a unique public key certificate in each card makes MULTOS incredibly flexible. For example, a bank can source its cards from multiple card vendors without having to exchange any personalisation keys with those vendors. It also makes instant issuance simple and secure. It is also possible to update cards in the field (say with a new loyalty app) using a POS terminal, ATM, NFC enabled smartphone or a PC.

17.4.3 Loading

For the moment, we will skip over the details of how MULTOS applications are personalised (that is, have user-specific data added to each instance of an application) and assume that we are going to load an application to a card that is fixed. A good example of this is the "Payment Systems Environment" application which is used by payment terminals to discover which payment applications exist on a card.

As we shall see, with MULTOS (and even step-one) the method for loading any application to a card is the same regardless of the application, MULTOS implementation or the manufacturer of the card. In fact, just one loading script (a sequence of APDU commands) needs to be implemented on a card personalisation machine for MULTOS. With other technologies, this script has to be customised for each application and implementation.

17.4.3.1 Loading

It is first necessary to look at the structure of a MULTOS Application Load Unit (ALU). It is helpful to think of it as a packet of data that is assembled off-card by a "data preparation" system. There are three variants as shown in Fig. 17.4.

In Fig. 17.4, security increases from left to right. The amount of security to be applied should be chosen to match the sensitivity of the application. It is tempting to treat everything with the highest security, but of course that has a cost in terms of loading performance as cryptography is involved.

For a Plaintext ALU, the only protection is a SHA-1 [1] hash of the application's code (held as part of the load certificate—more later).

In a Protected ALU, the plaintext ALU is signed using the secret half of a key known as the "application provider's key". This key's purpose is to assure integrity and authenticity of the application. It can also be thought of as the "application signer's key" as often the entity generating the signature is not the original application developer but instead a trusted third party performing the data preparation step (although the SHA-1 hash of the application code ensures that that is not at least changeable by a third-party ALU creator).

Finally, the Confidential ALU allows for some or all of the plaintext ALU to be enciphered for confidentiality. It does this by using a data element known as a Key Transformation Unit. This is encrypted by the card's public key and contains a map (offset/length pairs) of the data to be enciphered and a random symmetric key used to

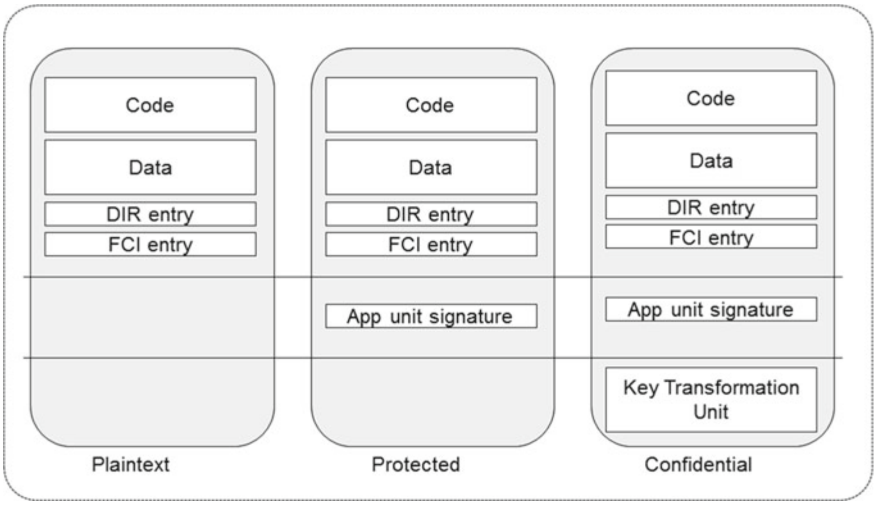


Fig. 17.4 Types of Application Load Unit

perform that encryption. The KTU therefore enables MULTOS to make use of PKI whilst operating as quickly as possible in-card by using symmetric cryptography for the bulk of the decryption work.

So, back to the actual loading process. This is very simply a case of pasting the formatted ALU data into the card, chunk at a time. A series of APDU commands exist for this (e.g. Load Code, Load Data, Load DIR etc.). It is trivial to write a script to divide up any ALU and send it in chunks to the card. However, this is not the end of the story. There has to be some control over the loadingof applications and this is achieved by the use of Application Load Certificates (ALCs).

17.4.3.2 Application Load Certificates

Looking back to Fig. 17.3, you will see that an input to the loading process is a certificate. This is required for any type of ALU. Before a new application (or version of an application) can be loaded to live MULTOS cards, it must first be registered at the KMA by the application developer. The data registered for each application includes (amongst other things)

- Its AID
- The size of the code segment
- The size of the static data segment
- The amount of session data used
- The size of the DIR and FCI entries
- A hash of the code segment

- The application's permission set for allowing access to certain features of MULTOS
- The type of the application (applications can run in several ways)

Once registered, an Issuer or approved agent (such as a card bureau) can obtain, via the KMA, a certificate for the registered application which additionally includes the ID of the issuer and crucially the public key of the application provider/signer.

Returning to the loading process then, we have got to the stage where the ALU has been pasted into the memory of the MULTOS chip. At this stage, the application cannot be executed because the load has not been authorised (and in fact in the case of Confidential ALUs cannot be completed). The final step is to send the relevant ALC to the card. Internally, MULTOS performs (amongst others) the following actions to confirm authorisation of the load.

1. Decipher the certificate using the KMA public key stored in the chip
2. Confirm the Application ID in the ALC matches that of the ALU
3. Confirm the Issuer ID in the ALC matches that added during enablement (see Sect. 17.4.2)
4. Confirm the code hash in the ALC matches the computed hash of the code in the ALU
5. Confirm the application sizes in the ALC match those of the ALU
6. Decipher any enciphered portions of the ALU (using the key recovered from the KTU)
7. Re-compute the hash of the plaintext ALU and confirm it matches the hash recovered (using the public key in the ALC) from the ALU signature

Only once the card has passed all the checks does it "know" that the application is genuine, unaltered and has permission to be loaded.

Note that it is possible to obtain ALCs from the KMA which will only work on specific MULTOS chips. In this case the ALC contains the serial number of the target chip. This is useful in post-issuance situations but in a bureau it would be impractical and of little value to have such card-specific certificates. It is also possible to make an ALC "single use" such that a card is only allowed to load a given application version once.

It is quickly worth mentioning Application Delete Certificates (ADCs). These are in effect, the reverse of ALCs. They are needed to be able to remove an existing application from a card.

17.5 Application Development

17.5.1 SmartDeck, the Application Development Environment

In 2015 the MULTOS development environment, SmartDeck [2], received a major overhaul. Previously the environment was not integrated, requiring the use of separate editing, build and debugging tools. In the latest incarnation, which this chapter describes, the edit, build and debug features have been incorporated into the Eclipse C/C++ Developer Toolkit framework. This not only makes the whole process easier, faster and more flexible, but also makes the many features of a modern IDE, such as Eclipse, available to the MULTOS application developer.

The main components of SmartDeck are:

- A toolchain plugin for Eclipse C/C++ Developer Kit
  - Provides compile, link and make functionality
- Compiler/Assembler/Linker command line tools
  - *hcl*, *hcc*, *has* and *hld*
- Simulator with debugger interface for Eclipse
  - *hsim* and *mdb*
- Standard ‘C’ libraries and header files
  - As used by the compiler
- Some helper macros for things like cryptography

The process flow is fairly standard and closely matches what you’d expect to see for ‘C’ programming. In Fig. 17.5 the shaded tools are command line tools called by

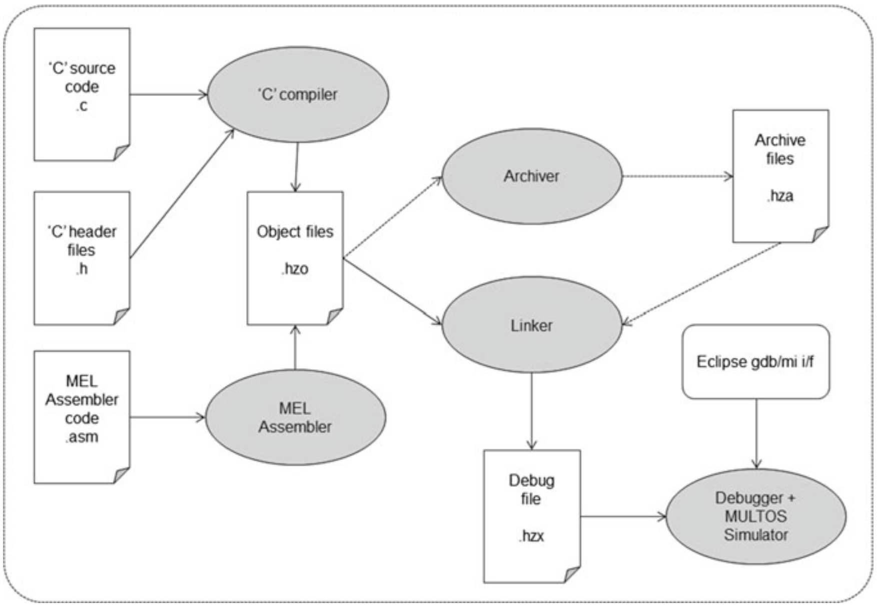


Fig. 17.5 Development process flow

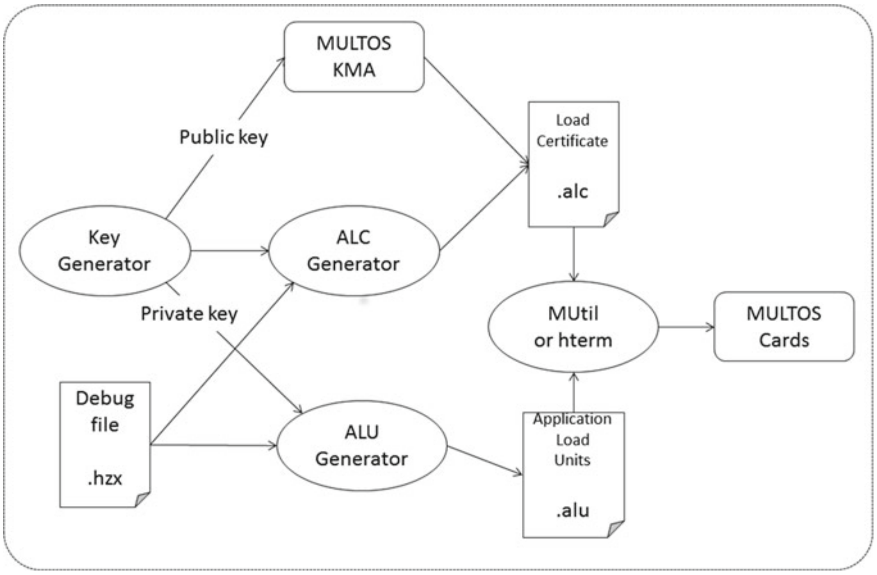


Fig. 17.6 Loader tools for development

the Eclipse plugin for MULTOS. The debug file contains all the static data, symbols and code references required for debugging. SmartDeck includes a default simulator, but the architecture allows for other simulators to be used (although this is really only useful if you are developing your own implementation of MULTOS).

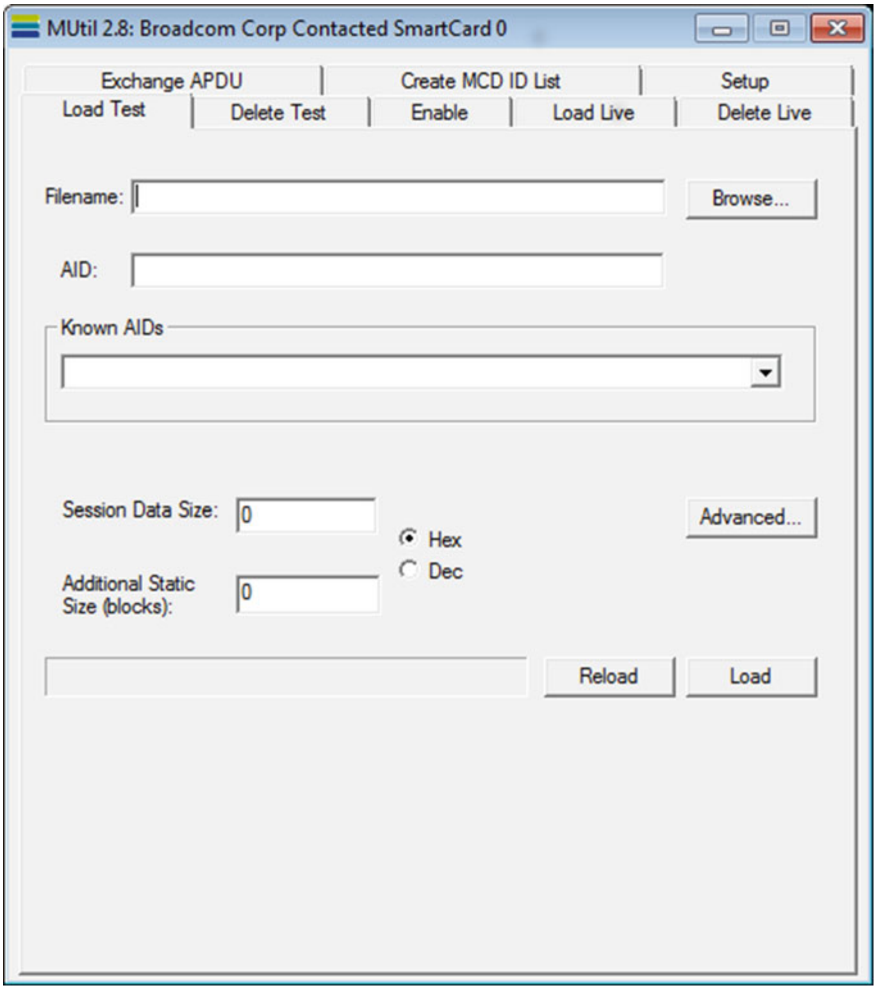
SmartDeck also includes a number of other tools (see Fig. 17.6) including:

- Key generator tool—*hkeygen*
- ALU generator—*halugen*
- ALC generator for test cards—*melcertgen*
- Object File lister (disassembler)—*hls*
- Binary file dump tool—*meldump*
- Terminal emulator *hterm*

These tools can be included in post-build scripts. A separate tool, MUtil [3] is also really useful when working with MULTOS cards.

The ALU Generator can generate personalised plaintext, protected and confidential ALUs (see Fig. 17.4). There is usually one ALU per application per card (i.e. personalised for each person).

ALCs (and in fact ADCs) are usually generated one per application, per issuer per card type. This gives the issuer control over which cards applications are allowed on. However, it is perfectly possible to control the loading and deleting of applications at an individual card level. This is especially useful for post-issuance changes to the card. Note that the MULTOS KMA must be used for obtaining certificates for live cards as the ALC generator only knows the required keys for developer (test) cards.



Filename:

AID:

Known AIDs

Session Data Size: 0

Additional Static Size (blocks): 0

[✓ Hex] Dec

Fig. 17.7 MUtil load test tab

MUtil (on the Load Test tab—Fig. 17.7) can generate unprotected ALCs for developer cards without the ALC generator tool and directly load applications. *hterm* can also load *hzx* files directly to developer cards.

17.5.2 *The Fundamentals of Writing a MULTOS Application*

Smartcard applications are very linear. In MULTOS, the operating system creates an instance of the application when selected and passes commands from the IFD.

The application performs some processing and then returns some data. To recap, the “static” memory is the hard-drive space of the application, and is persistent whatever happens. The “session” memory is persistent whilst the application remains selected until the chip is reset or powered down. However, each run of the application to process an incoming command is totally independent and has a new stack created for it.

The processing of a command (in ISO7816-4 APDU format [4]) usually follows a sequence similar to this:

1. Check the CLA byte is valid for your application
2. Check the INS byte is valid for your application (often with a switch statement)
3. Call CheckCase for the INS (more of CheckCase later)
4. Execute the code related to the instruction
5. Set any data to be returned by copying it to the public memory area
6. Set the status word
7. Exit

An APDU command may have one of the following formats (each element is one byte. The ‘|’ character signifies their concatenation).

Case 1: CLA | INS | P1 | P2
Case 2: CLA | INS | P1 | P2 | Le
Case 3: CLA | INS | P1 | P2 | Lc | Data
Case 4: CLA | INS | P1 | P2 | Lc | Data | Le

Where:

- CLA = Class byte: 0x00 denotes a standard ISO command
- INS = Instruction byte
- P1 and P2 = Parameter bytes
- Lc = Length of Data*
- Le = Length of expected response *

For example, the command to select the directory file is (in hexadecimal)

```
00 A4 00 00 02 2F00
```

* Where supported, these values can be more than one byte long, but that is rare. Note that **multoscomms.h**, one of the SmartDeck include files, defines macros for the different exit conditions corresponding to the different cases.

The *CheckCase* primitive (usually wrapped as a ‘C’ macro) is very important. It tells the O/S how to interpret the command data sent in the public memory area. Specifically whether Lc, Le or both are set. It validates that the data in the command matches the ISO case and sets the Lc and Le values.

Below is the code for the eLoyalty application included in SmartDeck. It is a VERY simple example that shows the basic structure. It allows points to be added and removed from the card and the points balance to be queried. It uses the libraries within SmartDeck. Actual code is in bold text.

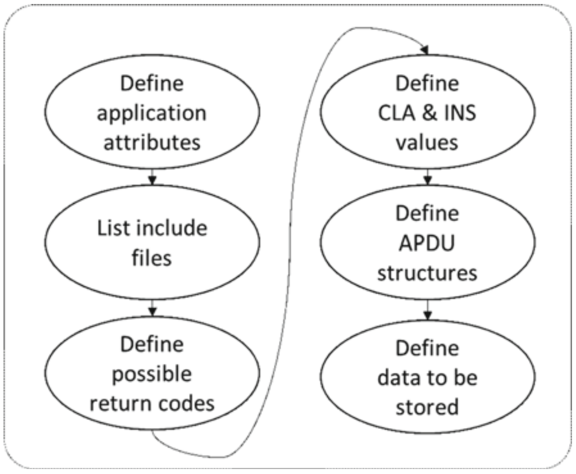


Fig. 17.8 Pre-amble to main()

17.5.3 Application Pre-amble

Before the actual application main() function a certain amount of pre-amble is required as shown in Fig. 17.8.

17.5.3.1 Define Application Attributes

```
#pragma attribute("aid", "f0 00 00 01")
#pragma attribute("dir", "61 10 4f 4 f0 00 00 01  50 8 65 6c 6f
79 61 6c 74 79")
```

The **#prama** compiler directive is used to set attributes understood by the SmartDeck tools. Here it is being used to set the AID and directory file entry.

17.5.3.2 List Include Files

```
#include < multoscomms.h >
```

This is the SmartDeck include file for handling I/O. Note, if using the [C-API] then this include file is replaced by the definitions in <multos.h>. SmartDeck has the standard include files you would expect to see for the C programming language. See Sect. 17.5.7 for more details.

17.5.3.3 Define Possible Return Codes

```
#define ERR_OK  0x9000
#define ERR_WRONGCLASS  0x6402
```

```
#define ERR_BAD_INS     0x6404
#define ERR_UNDERFLOW   0x6406
```

These are the status words that can be returned from the application after processing an APDU. The include file <iso7816.h>lists all the ISO standard status words.

17.5.3.4 Define CLA and INS Values

```
#define CMD_ADDPTS      0x10  /* Add points to the card
*/
#define CMD_SUBPTS      0x20  /* Deduct some points from
                                 the card */
#define CMD_QRYPTS      0x30  /* Return the number of points
                                 currently on the card */
#define CMD_APPINFO     0x40  /* Returns info about the
                                 application */
#define CMD_QRYID       0x44  /* Returns the ID info */
#define CMD_SETID       0x42  /* Sets the ID info */
#define MYAPP_CLA       0x70
```

17.5.3.5 Define APDU Structures

In this example, the APDU data in public is highly structured using structures for each APDU instruction and a union to combine them in the public data area.

```
/* APDU structure for QRYID and SETID commands. */
typedef struct
{
 char surname[20];
 char otherNames[40];
}APDU_ID;

/* APDU structure for ADDPTS, SUBPTRS, and QRYPTS commands. */
typedef struct
{
 int points;
}APDU_PTS;

/* APDU structure for APPINFO */
typedef struct
{
 char info[17];
}APDU_APPINFO;
```

```
static APDU_APPINFO app_info = { "Loyalty         " };
```

The compiler directive *#pragma melpublic* indicates that the following variables are in public memory.

```
/* Data from APDU; this is placed at PB[0] */
#pragma melpublic
union
{
 APDU_ID id;
 APDU_PTS pts;
 APDU_APPINFO info;
} apdu_data;
```

17.5.3.6 Define Data to Be Stored

The compiler directive *#pragma melstatic* indicates the following variables are to be stored in non-volatile storage.

```
#pragma melstatic

/* Static data for application. */
static APDU_ID id = { "Curtis", "Paul Lyndon" };
int points = 100;
```

17.5.4 Application Main() Function

This follows the pattern set out in Sect. 17.5.2. ExitSW (and its variants) cause the application to exit and return control to the MULTOS O/S. Each instruction has its own case statement.

```
void
main(void)
{
 int newPoints;

 /* Check class in APDU. */
 if (CLA != MYAPP_CLA)
  ExitSW(ERR_WRONGCLASS);
```

```
/* Decode instruction. */
switch (INS)
 {
 case CMD_ADDPTS:
  /* Ensure case 3 command. */
  if (!CheckCase(3))
   ExitSW(ERR_WRONGCLASS);

  /* Award points. */
  newPoints = points + apdu_data.pts.points;
  break;

 case CMD_SUBPTS:
  /* Ensure case 3 command. */
  if (!CheckCase(3))
   ExitSW(ERR_WRONGCLASS);

  /* Deduct points. */
  newPoints = points - apdu_data.pts.points;
  break;

  case CMD_QRYPTS:
   /* Ensure case 2 command. */
   if (!CheckCase(2))
    ExitSW(ERR_WRONGCLASS);

   /* Report points. */
   apdu_data.pts.points = points;
   ExitLa(sizeof(APDU_PTS));
   break;

  case CMD_QRYID:
   /* Ensure case 2 command. */
   if (!CheckCase(2))
    ExitSW(ERR_WRONGCLASS);

   /* Report ID. */
   apdu_data.id = id;

   /* Exit */
   ExitLa(sizeof(APDU_ID));
   break;

   case CMD_SETID:
    /* Ensure case 3 command. */
```

```
        if (!CheckCase(3))
         ExitSW(ERR_WRONGCLASS);
        id = apdu_data.id;

        /* Exit */
        Exit();
        break;

       case CMD_APPINFO:
        /* Ensure case 2 command. */
        if (!CheckCase(2))
         ExitSW(ERR_WRONGCLASS);
        apdu_data.info = app_info;

        /* Exit */
        ExitLa(sizeof(APDU_APPINFO));
        break;

       default:
        ExitSW(ERR_BAD_INS); // Invalid INS value
      }

      /* Common processing for point change. */
      if (newPoints < 0)
       ExitSW(ERR_UNDERFLOW);
      points = newPoints;

   /* Default exit status is 0x9000 */
   }
```

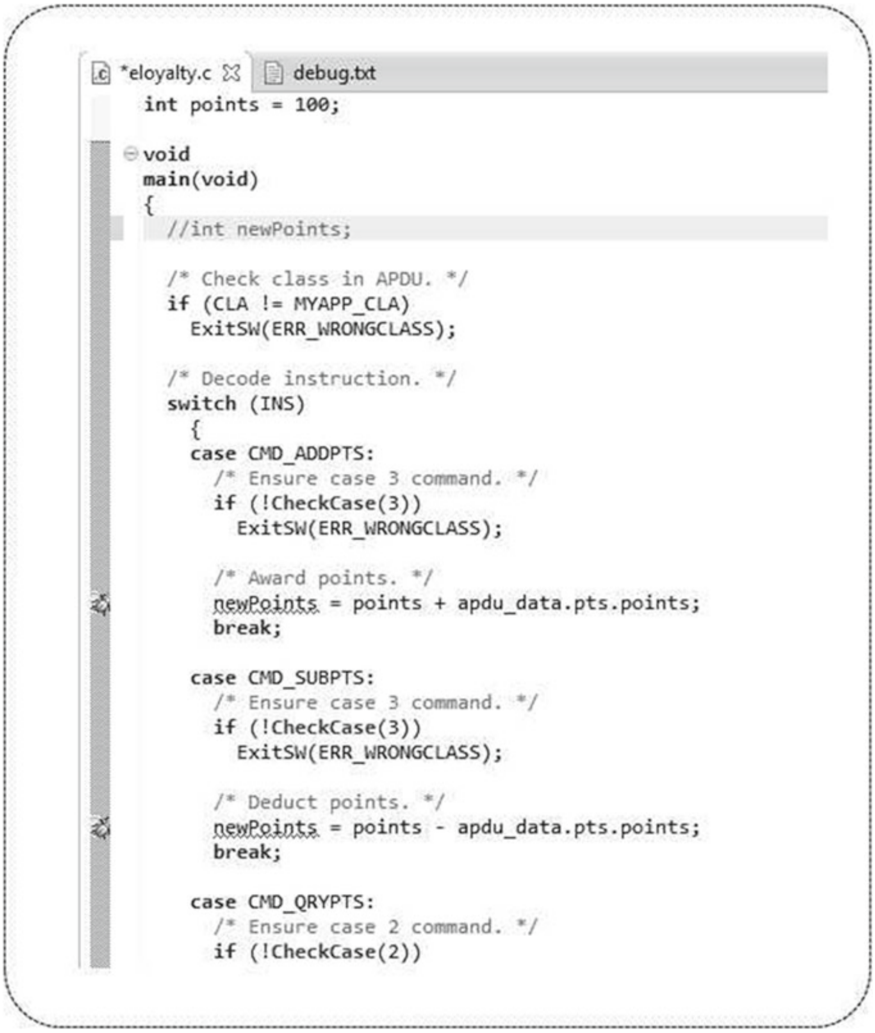
From the above example, you can see how essentially the basic command handling process is simple to implement.

17.5.5 Using the Eclipse IDE

Eclipse provides many features considered standard in a modern IDE including:

- Error detection
- Code completion suggestions
- Macro expansion

For example, in the above code example, if an error is introduced by commenting out the variable *newPoints* Eclipse automatically highlights the resulting errors. See Fig. 17.9.

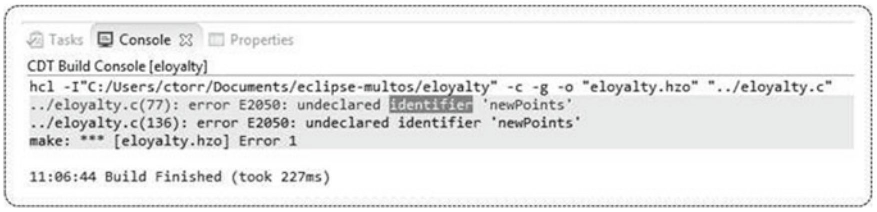


```
int points = 100;

void
main(void)
{
  //int newPoints;

  /* Check class in APDU. */
  if (CLA != MYAPP_CLA)
    ExitSW(ERR_WRONGCLASS);

  /* Decode instruction. */
  switch (INS)
    {
    case CMD_ADDPTS:
      /* Ensure case 3 command. */
      if (!CheckCase(3))
        ExitSW(ERR_WRONGCLASS);

      /* Award points. */
      newPoints = points + apdu_data.pts.points;
      break;

    case CMD_SUBPTS:
      /* Ensure case 3 command. */
      if (!CheckCase(3))
        ExitSW(ERR_WRONGCLASS);

      /* Deduct points. */
      newPoints = points - apdu_data.pts.points;
      break;

    case CMD_QRYPTS:
      /* Ensure case 2 command. */
      if (!CheckCase(2))
```

Fig. 17.9 Eclipse error detection

Eclipse can also be configured to interpret compiler error messages. The example workspace installed with SmartDeck has this set up already. Errors are highlighted in the console window (Fig. 17.10) and can be clicked on making the code editing window jump to the error location.

Documenting the generic functionality of Eclipse is outside of the scope of this chapter so we shall move on to debugging applications.



CDT Build Console [eloyalty]

```
hcl -I"C:/Users/ctorr/Documents/eclipse-multos/eloyalty" -c -g -o "eloyalty.hzo" "../eloyalty.c"
../eloyalty.c(77): error E2050: undeclared identifier 'newPoints'
../eloyalty.c(136): error E2050: undeclared identifier 'newPoints'
make: *** [eloyalty.hzo] Error 1

11:06:44 Build Finished (took 227ms)
```

Fig. 17.10 Compiler error messages in Eclipse console

17.5.6 Debugging MULTOS Applications with Eclipse

Firstly, each project requires a text file called, for example, “debugging.txt”. The first line of the file is the full path to hsim.exe (or indeed any MULTOS simulator that supports the required interface). The second line should be the AID of the application to be debugged. Subsequent lines of the file are the command line parameters you wish to use with the simulator. For **hsim** these are the usual ones to select the application and send APDUs. For example, the file may look like this:

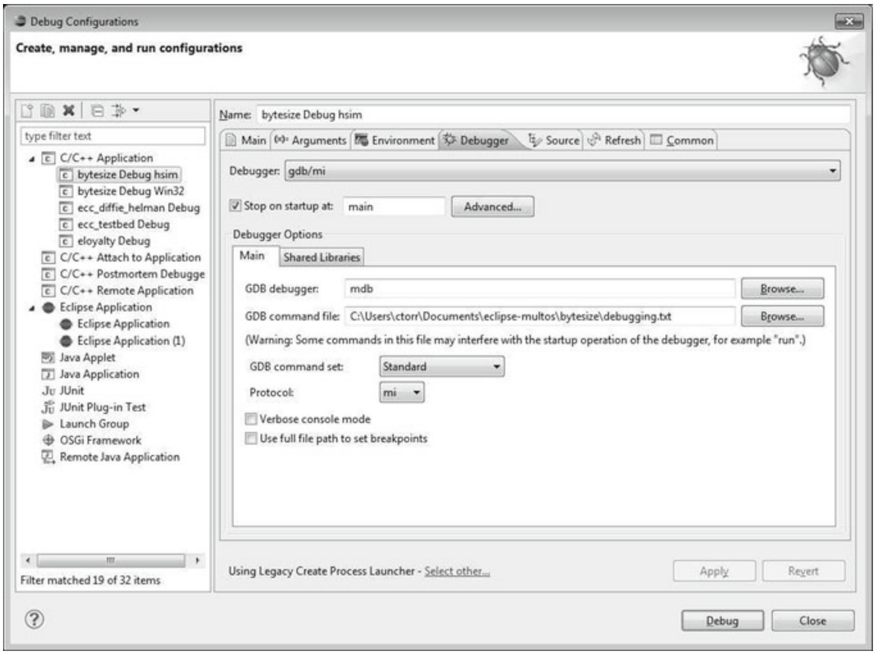
```
c:\program files (x86)\smartdeck\bin\hsim.exe
f3000003
-selectaid f3000003
-apdu 7001000002
-apdu 7002000002
```

You could include any of the other switches supported by *hsim*, but the name of the debug **.hzx** file is passed automatically, along with the switches required to support remote debugging.

To change the APDU sequence when debugging, you simply need to edit and save this file. Other simulators may use other mechanisms for application selection and APDU entry. APDUs can also be entered via the console window as we shall see later. This debugging instructions file is then used in the debug configuration for the project within Eclipse. Details of setting this up are given in the *MULTOS SmartDeck Manual* but an example is shown in Fig. 17.11.

Once configured, you press the debug button to start the debugging session. The application automatically breaks at main() at the start of each APDU command executed. The commands and options for stepping, setting breakpoints, running etc. are all defined by Eclipse. In fact, Eclipse has no idea that it is “talking” to MULTOS, it “thinks” it is just executing a ‘C’ application. It is not necessary to describe the details of Eclipse here, but it is worth pointing out the various features of Eclipse that SmartDeck makes use of.

So, to start with, the IDE switches to a debugging view. This allows you to have a different set of windows available for debugging as opposed to code development.



C/C++ Application
- bytesize Debug hsim
- bytesize Debug Win32
- ecc_diffie_helman Debug
- ecc_testbed Debug
- eloyalty Debug

C/C++ Attach to Application
C/C++ Postmortem Debugge
C/C++ Remote Application
Eclipse Application
- Eclipse Application
- Eclipse Application (1)

Java Applet
Java Application
JUnit
JUnit Plug-in Test
Launch Group
OSGi Framework
Remote Java Application

Filter matched 19 of 32 items

Name: bytesize Debug hsim

Debugger: gdb/mi

[✓] Stop on startup at: main

GDB debugger: mdb

GDB command file: C:\Users\ctorr\Documents\eclipse-multos\bytesize\debugging.txt

(Warning: Some commands in this file may interfere with the startup operation of the debugger, for example "run".)

GDB command set: Standard

Protocol: mi

[ ] Verbose console mode

[ ] Use full file path to set breakpoints

Using Legacy Create Process Launcher - Select other...

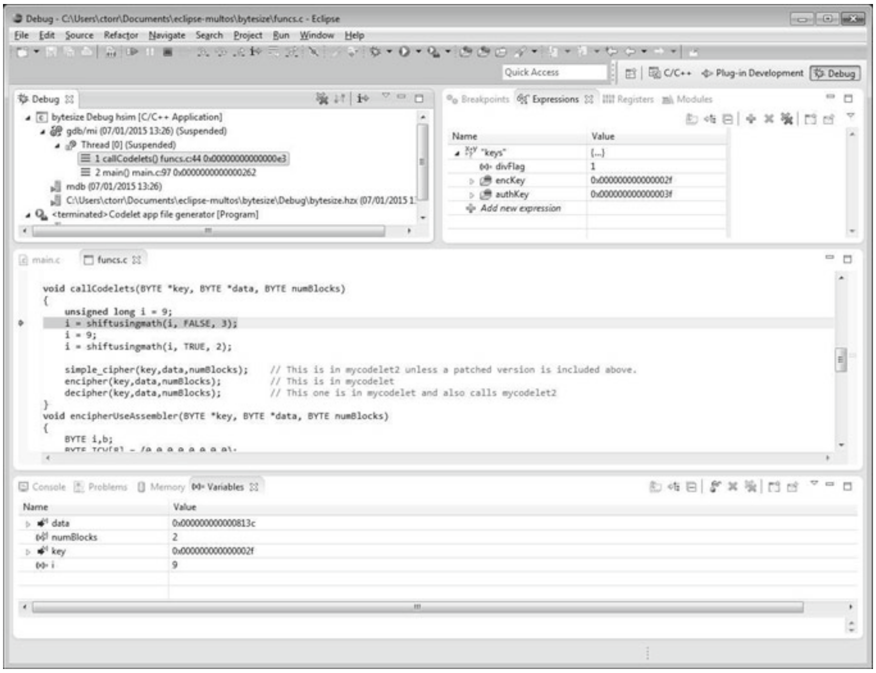
Fig. 17.11 Debug configuration

The layout of the view is fully customisable. The following screenshots show the various features supported by the MULTOS environment.

In Fig. 17.12 four windows are shown. In the top left is “Debug”. This shows the details of the process being debugged. At the top, the text “bytesize Debug hsim” is the actual name given to the debugging configuration. “gdb/mi” refers to the interface between Eclipse and this simulator. For MULTOS, the application mdb.exe replaces the GNU debugger gdb.exe. The actual MULTOS application is then shown under “Thread 0” (MULTOS devices, at the time of writing, are single threaded). Finally then in this window, you see the call stack of the running application with the code pointer value shown in brackets. You can click on the function shown in the call stack to switch the debugging context to that point. When you do this, the code window jumps and any local variables displayed change.

The top right window has the tab “Expressions” shown. The expressions window is used to primarily view variables that are not local ones or parameters. You add and remove the variables you want to watch using the buttons on the right hand side of the window. Clicking on a variable shows its full value in the pane to the right (blank in the screenshot) and structures can be expanded.

The central window shows the code being stepped through. Here you can set and clear break points and can hover over variable names to evaluate their value. It is also possible to step through assembler code in **.asm** files, with some limitations. Eclipse, of course, has no idea about the syntax of the file so cannot display anything in the


Debug - C:\Users\ctorr\Documents\eclipse-multos\bytesize\funcs.c - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
C/C++
Plug-in Development
Debug
Breakpoints Expressions Registers Modules
main.c funcs.c
Console Problems Memory Variables

```
bytesize Debug hsim [C/C++ Application]
  gdb/mi (07/01/2015 13:26) (Suspended)
    Thread [0] (Suspended)
      1 callCodelets() funcs.c:44 0x00000000000000e3
      2 main() main.c:97 0x0000000000000262
  mdb (07/01/2015 13:26)
  C:\Users\ctorr\Documents\eclipse-multos\bytesize\Debug\bytesize.hzx (07/01/2015 1
<terminated>Codelet app file generator [Program]
```

Name	Value
"keys"	{...}
divFlag	1
encKey	0x000000000000002f
authKey	0x000000000000003f
Add new expression	

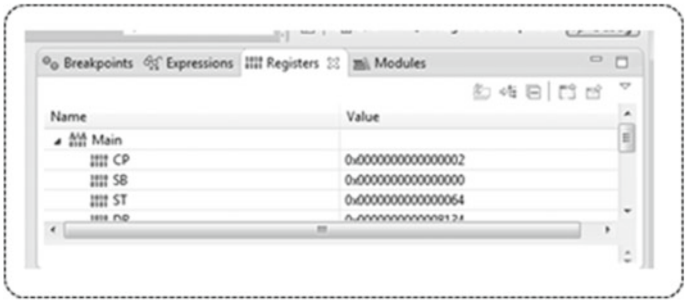
```
void callCodelets(BYTE *key, BYTE *data, BYTE numBlocks)
{
    unsigned long i = 9;
    i = shiftusingmath(i, FALSE, 3);
    i = 9;
    i = shiftusingmath(i, TRUE, 2);

    simple_cipher(key,data,numBlocks);    // This is in mycodelet2 unless a patched version is included above.
    encipher(key,data,numBlocks);         // This is in mycodelet
    decipher(key,data,numBlocks);         // This one is in mycodelet and also calls mycodelet2
}
void encipherUseAssembler(BYTE *key, BYTE *data, BYTE numBlocks)
{
    BYTE i,b;
```

Name	Value
data	0x000000000000813c
numBlocks	2
key	0x000000000000002f
i	9

Fig. 17.12 A debugging session

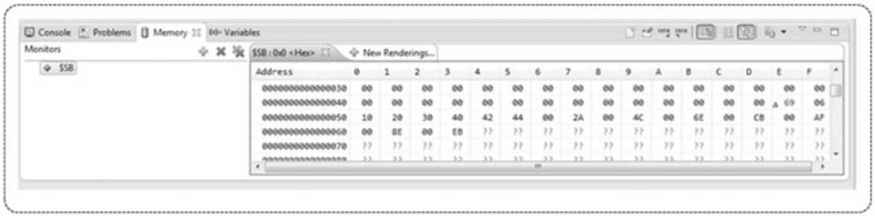
"Variables" window. You can still however evaluate any variable in the "Expressions" window and use the "Memory" window to view memory being used by the assembler code. The bottom window is showing the "Variables" tab. This lists the local variables and function parameters. You can click on items in this list to evaluate them or drill into them.

Other debugging windows allow you to watch the MULTOS registers (Fig. 17.13) and monitor memory (Fig. 17.14). For the latter you can enter a memory address



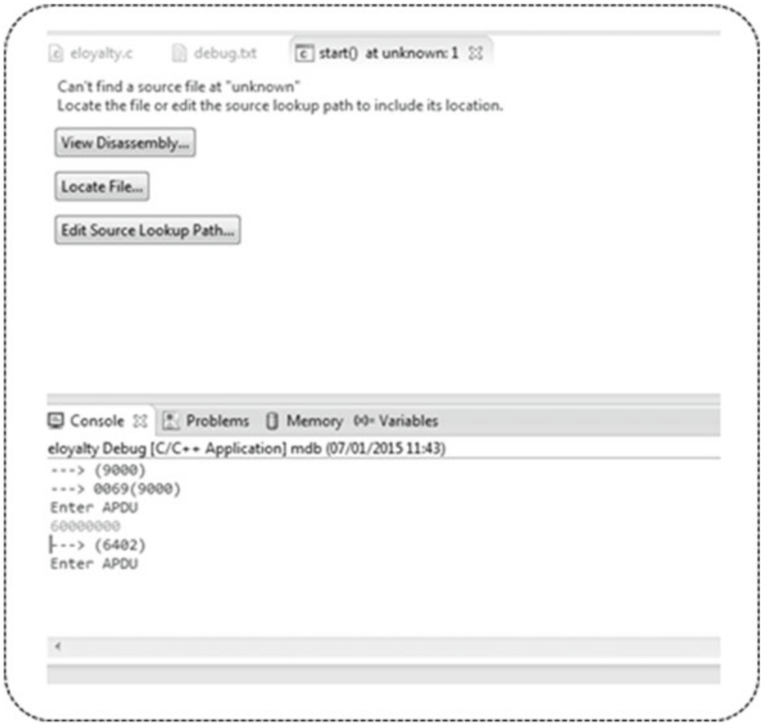
Name	Value
Main	
CP	0x0000000000000002
SB	0x0000000000000000
ST	0x0000000000000064

Fig. 17.13 Monitoring MULTOS registers



Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000000000000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0000000000000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	69	06
0000000000000050	10	20	30	40	42	44	00	2A	00	4C	00	6E	00	CB	00	AF
0000000000000060	00	8E	00	EB	??	??	??	??	??	??	??	??	??	??	??	??
0000000000000070	??	??	??	??	??	??	??	??	??	??	??	??	??	??	??	??

Fig. 17.14 Monitoring memory addresses



Can't find a source file at "unknown"
Locate the file or edit the source lookup path to include its location.

eloyalty Debug [C/C++ Application] mdb (07/01/2015 11:43)

```
---> (9000)
---> 0069(9000)
Enter APDU
60000000
|---> (6402)
Enter APDU
```

Fig. 17.15 mdb console

(e.g. 0x8000), the name of a pointer variable (e.g. ptr), the address of a variable (&myvar) or an address pointed to by one of the registers (e.g. $PB).

You will notice in Fig. 17.14 that some memory locations are given the value **??**. This is, if you remember, because the amount of memory allocated to an application is set at the time it is loaded. **??** shows memory addresses that are not valid for the running application. Also, note that changed values are highlighted in red (or a colour of your choice) as you step through your code.

Other very useful windows are the “Console” windows. These present the *stdin*, *stdout* and *stderr* streams of the various levels of application running during debugging. The most important of these is the Console window for *mdb* (Fig. 17.15). From here you can see the output of your MULTOS application and can inject additional APDU commands once those in the "debugging.txt" file have been exhausted.

You can also see in Fig. 17.15 what is displayed in the code window when the application is not executing an APDU.

17.5.7 Using the Standard C-API for MULTOS

SmartDeck comes with its own set of header files and libraries that cover the basic functionality as follows:

- multi-precision unsigned arithmetic in <multosarith.h>
- access to the Code Condition Register (CCR) in <multosccr.h>
- cryptographic primitives in <multoscrypto.h>
- controlling communications with the IFD in <multoscomms.h>
- other MAOS services in <multosmisc.h>
- a simulator-only printf capability in <stdio.h>
- non-local jump support in <setjmp.h>
- memory functions in <string.h>
- pseudo heap management functions in <stdlib.h> and <heap.h>
- DES and RSA encryption in <DES.h> and <RSA.h>
- standard C definitions in <stddef.h> and <limits.h>

However, the standard C-API, available to download (multos.h) provides access to all the latest primitives and many of the features in the SmartDeck header files. You may wish to consider making use of the C-API instead.

17.5.8 Different Types of MULTOS Application

At the time of writing there are three types of MULTOS application as follows:

- **Normal** the application has to be selected explicitly and only processes application commands.
- **Default** the application is selected automatically after a reset, and again only processes application commands.
- **Shell** the application is selected automatically after a reset AND has to handle ALL incoming commands.

Shell applications can be useful in highly non-standard environments but need to be programmed carefully to avoid putting the chip into a locked state. Default applications are useful for embedded applications to simplify the boot process or in environments where the terminal does not send select commands.

17.5.9 Personalising MULTOS Applications

Personalisation is the process whereby a generic application is populated with data that is specific to the user of that application. In the simplest case, it could be a person's name. Typically though there are many items of data that need to be personalised and much of it is secret (like PIN codes and cryptographic keys).

One very common approach to this is to give the application a set of APDU commands that allow a secure channel (using symmetric keys) to be established and then to send other APDU commands, via that channel, to set the personalised data items in the card (either in blocks, or one at a time). This has several drawbacks including the need to securely distribute and store (in HSMs) many symmetric keys and for numerous cryptograms to be calculated on-the-fly whilst a card is in the chip programming head of a personalisation machine (which also potentially lengthens the process as well as consuming HSM resources).

The usual MULTOS approach, although you are still able to use the approach mentioned above if you wish to, is to populate the personalised data into application templates prior to the data reaching the personalisation machine. Referring to Fig. 17.4 then, this offline personalisation process pokes values into the relevant addresses of the Data part of the ALU which can then be encrypted via the KTU mechanism. The KTU can in turn be encrypted directly by the card key (in which case it is ready to load to the card) or be encrypted under a temporary transport key (in which case a single HSM call is needed on the personalisation machine to re-encipher the KTU from the transport key to the card public key).

This has several important benefits:

- You don't have to know anything about a card and its keys before you start to personalise it. You simply ask the card for its public key certificate which tells you everything you need to know.
- All the slow calculations can be done up-front, saving time on the personalisation machines, where it is simply a process of pasting in the ALU.
- Except for potentially one HSM call, the MULTOS card does all the cryptography required internally using an optimised method.
- Using the offline method of application personalisation allows you to have a single script on your personalisation machines for personalising ANY MULTOS application on ANY MULTOS card. This saves a huge amount of time and money in script development time.
- In an instant issuance scenario, there is no need for the remote issuing site (be it a bank branch or kiosk) to have an HSM, as all the data transmitted to the card is already fully cryptographically protected to ensure authenticity, integrity and confidentiality. In fact, a totally open transmission channel can be used.

Note that the SmartDeck tool *halugen* allows you to personalise applications. See the *MULTOS SmartDeck Manual* for details.

17.5.10 Future Uses for MULTOS

MULTOS is an operating system for a secure microcontroller. Traditionally that has been for a very specific kind of microcontroller with very limited (by design) input/output capability. However there is no reason to restrict MULTOS to such devices. Secure microcontrollers with other I/O options do exist and could be used to control and secure all manner of connected devices in the market very loosely labelled “The Internet of Things”, or IoT for short. At the time of writing[1] at least one such MULTOS product is being developed.

For complex devices, a MULTOS chip could play the role of a secure co-processor; in more simple devices MULTOS could be the O/S of the main CPU providing control, secure processing, secure storage and communications. The biggest advantage of using MULTOS in such devices is the ability to remotely deploy applications (usually termed firmware in this context) to devices in a number of interesting and secure ways. For devices that are probably going to be in the field for upwards of ten years, confidence is needed that they will remain secure and have the ability to be upgraded during their lifetimes.

17.6 Summary

This chapter on MULTOS has hopefully provided some insight into this purpose-built secure operating system for smart cards (and other devices). It has shown how easy it is to develop and deploy applications using the SmartDeck toolset and the Eclipse IDE framework and explained some of the benefits of choosing MULTOS for smart card based projects. All the tools mentioned are available to download for free from the MULTOS website www.multos.com. Information on obtaining sample cards can be obtained by e-mailing mailto:info@multos.com.

17.7 Bibliography

The MULTOS documents shown in Table 17.2 provide further in-depth reading and are available online at www.multos.com.

[1]May 2016.

Table 17.2 Useful resources

Document and Synopsis	Of interest to
Delegation: Describes how to delegate control from one application to another.	Application developers
Enablement: Briefly describes the enablement process, its purpose and how to do it.	Anyone interested in the security aspects of MULTOS
Delegation: Describes how to delegate control from one application to another.	Application developers
Shell Applications: Describes how to write applications that are selected by default and process all commands e.g. to mimic single application cards or to support terminals that do not send a Select Application command.	Application developers
MDG-MULTOS Developer's Guide: An introduction to smart cards in general. • A description of the MULTOS architecture. • Application development principles using lots of 'C' and MEL examples for common tasks.	Application developers
MDRM-MULTOS Developer's Reference Manual: Functional descriptions of • MULTOS APDUs • MEL Instruction set • primitives (indicating mandatory or optional and primitive sets)	Application developers
GLDA-Guide to Loading and Deleting Applications: Describes the elements involved in loading and deleting applications and how they inter-relate.	Anyone interested in the security aspects of MULTOS. Anyone involved in the delivery of applications or personalisation
GALU-Guide to Generating Application Load Units: Detailed description on how to create a personalised ALU.	Issuers, Personalisers and Security
MIR MULTOS Implementation Reports: Details the features of products on offer by different implementers.	Issuers and developers looking for MULTOS products to use
CAPI MULTOS Standard C API: Documents the 'C' interfaces available to MEL instructions and standard MULTOS primitives as found in the multos.h file available for download on the multos.com website.	Application developers
MUM MULTOS Utility Manual: A guide to the MUtil application available for download on the multos.com website.	Anyone developing MULTOS products
FIF MULTOS KMA File Interface Formats: This document is needed to integrate output from the KMA into personalisation tools.	Vendors, Issuers, Processors
MULTOS SmartDeck Manual: Describes in detail the SmartDeck tool set for building and debugging MULTOS applications including installation and use.	Application developers

References

1. NIST—FIPS180-4 Secure Hashing Standard. *One-way cryptographic hashing functions used for digital signatures.* Available via http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf Accessed on April 2016.
2. MAOSCO Ltd—SmartDeck. *Software development environment for MULTOS.* Available via https://www.multos.com/developer_centre/tools_and_sdk/ Accessed on April 2016.
3. MAOSCO Ltd—MUtil. *Graphical MULTOS Utility for card content management and command exchange.* Available via https://www.multos.com/developer_centre/tools_and_sdk/ Accessed on April 2016.
4. ISO—ISO/IEC 7816-4:2013. *Integrated circuit cards—Part 4: Organization, security and commands for interchange.* Available via http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=54550. Accessed on April 2016.

Chapter 18
Trusted Execution Environment and Host Card Emulation

Assad Umar and Keith Mayes

Abstract Over the years, mobile devices have become increasingly sophisticated in terms of their features and the use cases they operate. This rise in sophistication poses a major security threat because it increases the attack surface of mobile devices. Consequently, the challenge from a security point of view is to offer security assurances for applications and services hosted on these devices. In this regard, a Trusted Execution Environment (TEE) as a technology provides an execution and storage platform on the device, which is isolated from the rest of the operating system and other applications, and is intended to be trustworthy. This provides security assurances in terms of the confidentiality and integrity for applications and their related data, running on the TEE. In this chapter, we explore what constitutes a TEE and the various security features a TEE is expected to provide. We also highlight standardisation efforts relating to TEEs. Example implementations of TEEs are contrasted along with Host Card Emulation (HCE) used in Near-Field Communication (NFC). NFC card emulation has traditionally relied on a TEE in the form of tamper-resistant Secure Element (SE) chip, whereas HCE allows an application on the host CPU of the mobile device to emulate a smart card. HCE introduces new security risks and this chapter considers how these can be managed to an acceptable level.

Keywords Mobile devices · NFC · Security · Trusted Execution Environment · Secure Element · Platform integrity · Host Card Emulation · Tokenisation · Secure storage · Secure execution

A. Umar (✉)
Smart Card Centre, Information Security Group, Royal Holloway,
University of London, Egham, UK
e-mail: assad.umar.2011@live.rhul.ac.uk

K. Mayes
Director of the Information Security Group, Head of the School of Mathematics
and Information Security, Royal Holloway, University of London,
Egham, Surrey TW20 0EX, UK
e-mail: keith.mayes@rhul.ac.uk

K. Mayes and K. Markantonakis (eds.), *Smart Cards, Tokens, Security and Applications*, DOI 10.1007/978-3-319-50500-8_18

18.1 Introduction

It is important to appreciate the subtle difference between “trusted” and “trustworthy”. The word *trusted* in this context refers to any software or hardware component you depend on not to compromise the security of a device, while *trustworthy* refers to a component that will not compromise the security of a device, and hence it is “worthy of trust”. Execution environments typically consist of components such as a processor, memory, some form of storage, and other peripherals such as crypto-processors depending on the specific use case. A Trusted Execution Environment (TEE) therefore is a set of software and hardware components relied upon to provide a secure storage and execution environment on devices. The TEE, where applicable, forms the backbone of the device’s Trusted Computing Base (TCB). The TCB of a device represents a collection of all software, firmware, hardware components that are explicitly trusted to enforce security on a device. Typically, TEEs are logically separated from the Operating System (OS), applications, and other peripherals running in the normal execution environment of the device, thereby taking them out of the TCB. This provides the notion of “Trusted Execution Environment” and “Rich Execution Environment” (REE) for the trusted/secure and untrusted sides of the device, respectively.

As mobile devices become more sophisticated and “smart”, concurrently running different applications, the need for security increases tremendously. The need for security can be seen from different perspectives in many use cases. For example, the users want assurances on the privacy and security of their data. The Mobile Network Operator (MNO) on the other hand may want to ensure mobile devices have immutable IDs so they can lock users to their subsidised contracts. TEEs provide security features that contribute to maintaining acceptable levels of security on mobile devices. The boundaries between TEEs and the security features they provide are not always precise and have varying nomenclature, so Sect. 18.2 presents the main generic TEE features. It should be noted that not all of these features are found across the range of TEE implementations.

18.2 Trusted Execution Environments

The generic security features a TEE should provide are stated in [1]. They are outlined as follows:

- Isolated execution
- Secure storage
- Remote attestation
- Secure provisioning
- Trusted path

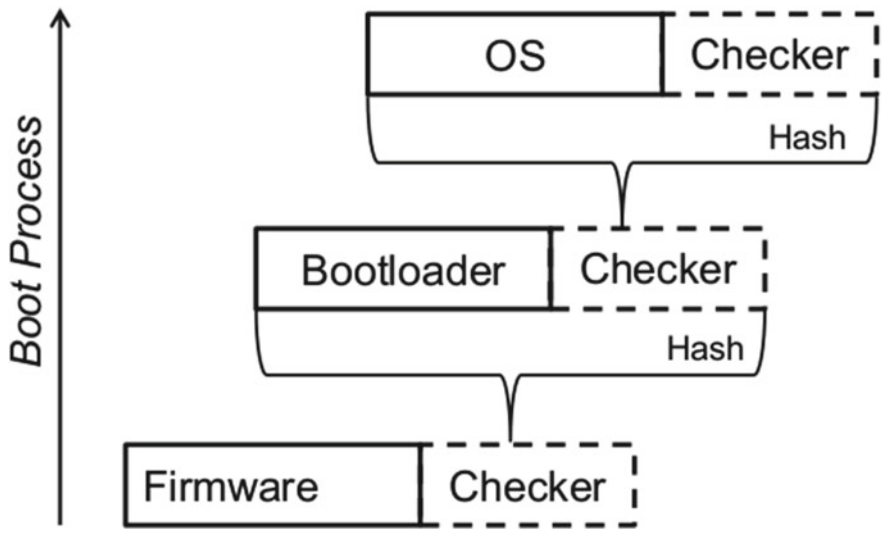


Fig. 18.1 The secure boot process

Before going into the security features in more detail, it is important to understand the idea of "trusted boot", because for all the aforementioned security features to hold, the software and configurations running on the devices must get to an initial trusted state.

Normally when any device is powered up, the firmware finds the boot block containing the boot loader, which then loads the OS. This is the generic way of booting (bootstrapping) any device. In trusted boot, the device is booted from a known and trusted good value. Although the terms are used interchangeably, there are two flavours of trusted boot: *secure boot* and *authenticated boot*.

In the secure boot, shown in Fig. 18.1, every stage is verified by comparing the value of the code at each stage of the boot process to a known good value. Typically, this is done by taking the hash of the code (checker) at that stage and comparing it to a hash value of a known and trusted piece of the code. If successful, the boot process continues to the next stage, otherwise it quits.

On the other hand, the authenticated boot (illustrated in Fig. 18.2) lets you boot any code you wish, but whatever is booted is stored in the form of a hash, so instead of a "checker" in the case of secure boot, the authenticated boot has a "measurer" at every stage of the boot process. The measurements are saved in the form of an aggregated hash. Authenticated boot therefore can be used to put some sort of access control policy framework on devices, so although the user is allowed to run the

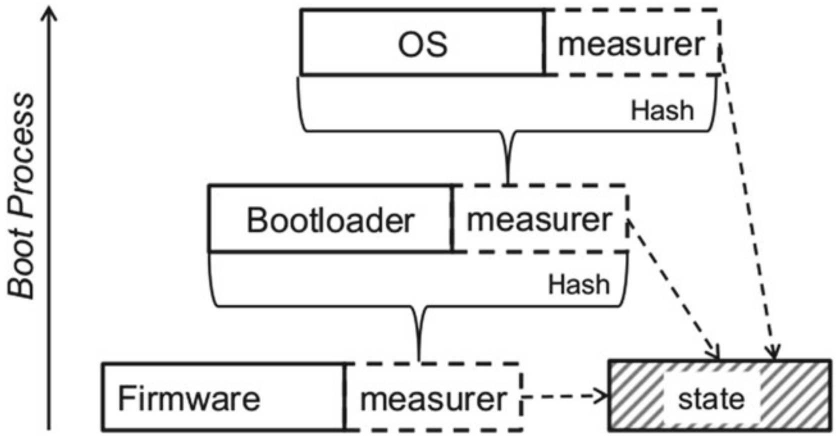


Fig. 18.2 The authenticated boot process

device from any configuration, the device must prove that it is running a particular configuration or version of software (attestation). In summary, secure boot only lets the device boot from a known value, whereas authenticated boot lets the device boot up from any software configuration, and the state measured. Authenticated boot is also referred to as measured boot in alternative nomenclature.

18.2.1 Isolated Execution

Isolated execution is the ability to execute a piece of code in complete isolation to the rest of the code running on the device including the main OS. This offers *confidentiality* and *integrity* of the code and related data at “run-time”. For example, a payment application on a mobile device should be able to generate digital signatures, referred to as cryptograms in EMV [2], without applications running in the REE observing the process or having access to the signing keys.

Traditionally, operating systems provide a notion of isolated execution; for example, Android uses *sandboxing* to ensure applications can only access data within their sandbox. However, this approach puts the OS in the TCB. The problem here is if a malware or rootkit attacks the OS, then sandbox protection is also vulnerable. Furthermore, having the OS as part of the TCB makes the size of the TCB huge, thus increasing its overall attack surface. Therefore to achieve “true” isolated execution, the main OS must be taken out of the TCB.

The security of processes running in TEE isolation should be assured even when the OS has been compromised. Isolated execution can be realised in a number of ways, including by having a separate co-processor to run the trusted applications in parallel to the processor running the rich OS. The Trusted Platform Module (TPM) (see Chap. 7) is an established example of using a separate module for isolation. Another way of realising isolation is through virtualisation, in which a hypervisor is put between the OS and the hardware, giving a single processor the ability to be in two different modes: trusted and untrusted. ARM's TrustZone [3] is a good example of using this type of approach. ARM refers to the two modes as "secure world" and "normal world".

18.2.2 Secure Storage

While the previous section focused on the security of application code at *run-time*, the integrity, confidentiality, and in most cases the freshness of data at rest must also be assured, for example, applications such as transit, payment make use of sensitive data such as cryptographic keys, passwords, and Primary Account Numbers (PANs) that must be protected against unauthorised access.

Normally, the OS provides a weak notion of secure storage based on permissions. Any available hardware tamper-resistant chip used for secure storage is known as the *Hardware Root of Trust for Storage (RTS)*. In an attempt to keep the TCB as small as possible, it is important to introduce the idea of *"storage sealing"*.

Storage sealing is the technique of securing an off-the chip storage by encrypting it with keys resident in an RTS, for example a TPM. This makes it possible to secure data in an insecure storage environment that is much bigger than the capacity of the RTS. This is also where *freshness* of the data comes into play as the state of the data in the sealed storage must be maintained otherwise an attacker can replace the encrypted data with an older copy. For example, blacklisting is widely used in access control and transport ticketing, assuming the blacklist is encrypted and stored in an insecure environment, an attacker might replace the encrypted blacklist (which contains the attackers ID) with an older version of the blacklist that does not include the attackers ID. Typically, freshness is provided by using a small amount of non-volatile memory as a counter or adding a time-stamping mechanism.

18.2.3 Remote Attestation

Attestation literally means "vouching for something". *Remote attestation* is the ability to vouch for an entity or its characteristics to a third party. It is used to give assurances to a service that a piece of software or OS is trustworthy. Thus, the remote entity has confidence that it is not interacting with a malicious entity such as a malware.

To contextualise this, in mobile payments, the issuer should have assurances for the integrity of the banking application on the mobile device. Another example could be Virtual Private Network (VPN) access to a university's network. To reduce the risk of exposure to attacks, the university may have a policy of only allowing access from a computer that can attest to the trusted state of its OS.

A remote attestation protocol will normally begin with an integrity measurement of the TCB as a whole. This is done by taking a digest of the code, say using a hash function, and signing it with a key, stored preferably in a hardware RTS. A third party can verify the signature with a corresponding public key and compare it against a list trusted OS hashes.

18.2.4 Secure Provisioning

Secure provisioning is the ability to send data to a device or a particular software component running on the device while ensuring its confidentiality and integrity. This mechanism is particularly important in the smart cards and mobile ecosystem because Service Providers (SPs) need a way to provision applications and other supportive data to their users' devices. For example, in payment applications, banks require a secure mechanism to deliver the application to the device of its user. There must be assurances on the integrity of the application as well as the confidentiality of other cryptographic data such as digital certificate. The user on the other hand is assured of the authenticity of the application and its origin. Another wide application of secure provisioning mechanisms is verifying updates such as firmware updates before installation. This is typically done by carrying out integrity checks such as hashes and digital signatures on external data.

18.2.5 Trusted Path

Trusted path, also referred to as trusted User Interface (UI), is a mechanism that allows a secure interaction between the user and trusted applications running on a device through its input/output devices, such as mouse, screen, and keyboard. To achieve security in a holistic manner across the system/device, secure storage and execution environment alone are not sufficient. A malware on the device may be able to access sensitive data entered by user, by logging keystrokes or by more sophisticated methods such as sniffing data on the physical link while in transit. Therefore, the trusted path is controlled by the TEE and is part of the TCB. Furthermore, a payment application, for example, may require the user to be authenticated using passwords and PINs. The TEE should provide a specific mode in which it controls the device and all the components required by the user to enter the password. Typically, this will involve a secure screen and keyboard.

18.3 Standardisation of TEEs

Standardisation of TEE is important to ensure interoperability among the different stakeholders involved and makes sure fragmentation across the ecosystem is minimised. Standardisation also forms the basis for a robust certification of TEEs to be in place. There are different standardisation bodies involved with TEEs and their efforts are discussed below.

18.3.1 Trusted Computing Group (TCG)

Predominantly, the Trusted Computing Group (TCG) was involved with Trusted Platform Modules (TPM) used in Personal Computers (PCs) and servers. In addition, they have standards (TPM v1.2 and TPM v2.0 mobile specifications) [4] for mobile devices under the *"Mobile Platform Work Group"*. The TPM itself is a hardware chip that provides secure services and also collects information about the system in a secure and trusted way. This information can then be used to give security assurances to a remote entity (remote attestation). The TPM v2.0 standards provides new features such as secure and measured boot, support for *algorithm agility*. Algorithm agility is the ability to switch between cryptographic algorithms should the one in use be deemed weak. At the time of writing[1], TPM v2.0 are the current standards and will now be explored in more detail.

18.3.1.1 TPM v2.0 Mobile Specifications

The TPM v2.0 mobile specifications are a "family" of three specifications: Mobile Reference Architecture [4], Mobile Common Profile [5], and Mobile Command Response Buffer Interface [6].

- **TPM 2.0 Mobile Reference Architecture**
  This specification does not impose a particular implementation for the protected environment. For example, possible implementations may include using a hypervisor, a dedicated core separate from the one used by the "rich" side of the device, a dedicated hardware chip, or other hardware isolation techniques [4]. The protected environment essentially provides an execution environment that executes "trusted applications" that are completely obscure to applications running on the host OS.
- **TPM 2.0 Mobile Common Profile**
  The mobile common reference defines the TPM's profile which is applicable to mobile devices that conform to the Mobile Reference Architecture.

[1] August 2016.

- **TPM 2.0 Mobile Command Response Buffer (CRB) Interface**
  This specifies an interface between the TPM on a device and a piece of software. This interface is known as the CRB interface, and it aims to work with any of the TPM architecture implementations mentioned previously. The CRB interface allows for device drivers to communicate with a TPM regardless of its implementation style. To communicate with the TPM, the firmware puts the command to be executed in the TPM's command buffer and sets a flag to *Start*, so that the TPM executes the command in the buffer. After a successful execution, the TPM puts the response in the *response buffer* and clears the *Start* flag [6].

18.3.2 Unified Extensible Firmware Interface (UEFI)

The Unified Extensible Firmware Interface (UEFI) is an industry initiative supported by many companies in the personal computing sphere; their efforts are geared towards PCs. However, Windows phones use the UEFI Secure Boot Feature Specification 2.2 [7]. The secure boot is a mechanism that launches only drivers and images that were originally signed by known and trusted keys. If the verification fails, the UEFI firmware starts a recovery process to restore a trusted version of the firmware.

The secure boot mechanism has three modes; *setup mode*, *user mode*, and the *custom mode*. If secure boot is enabled on a device, it starts in the setup mode. A public key, referred to as the *platform key*, is written to the firmware and then process goes into the "user mode". In user mode, only drivers signed by the platform key will be loaded. Custom mode allows users to modify the content of the *secure boot signature database* and also the platform key, thereby providing more flexibility. The UEFI specification version 2.6 can be found in [7].

18.3.3 GlobalPlatform TEE Specifications

GlobalPlatform is a non-profit organisation made up of companies and organisations, which develop specifications for the secure provisioning and management of applications on chip technology. Traditionally, GlobalPlatform was concerned with mainly smart cards; however, its efforts now extend to the mobile ecosystem. GlobalPlatform, through its "Device Committee", have come up with a number of specifications related to TEEs. We now discuss the most significant ones in more detail.

18.3.3.1 TEE System Architecture Specification

Although GlobalPlatform does not insist on any specific architecture, this specification [8] explains software and hardware configurations that make up a TEE. It also highlights various components that could be used to host a TEE, as well as resource sharing and memory management between the rich OS and the trusted OS.

TEEs typically require the same components as the REE, which include processing cores, cryptographic processors, memory, RAM, ROM, and other components that make up the platform chipset. The GlobalPlatform standards provides three broad hardware architectures that cover most existing TEE implementations. As depicted in Fig. 18.3, a TEE could be realised as an external chip as shown in (A), (B) shows a TEE that can be realised by using a dedicated chip embedded within the platform, or as (C) shows, a TEE could also be a logical separation of all the components into *secure* and *insecure* worlds, referring to the main platform as the System on Chip (SoC).

18.3.3.2 TEE Client API Specification

This specification [9] facilitates communication between applications running on the "rich OS" (referred to as client applications in the specification) and applications running on the TEE, also referred to as *trustlets*. The client API is a simple mechanism that opens a channel of communication and afterwards the rich and the trusted applications that deal with the semantics. Specific operations such as secure storage and cryptography may be built on top of the client API, but the details of that are outside the scope of the specification [9].

18.3.3.3 TEE Internal API Specification

This specification [10] allows for the development of trusted applications regardless of the underlying TEE implementation. The internal API should be suitable as long as the implementation conforms to the *"System Architecture Specification"*. This means that the TEE should be accessible by client applications in the rich OS.

18.3.3.4 TEE Trusted User Interface API

As mentioned in Sect. 18.2.5, secure storage and execution environments alone are not always sufficient. Certain scenarios may require a secure interaction between the user and the application. For example a user entering a PIN for verification, as widely used in mobile payment. The user may also be required to verify certain information displayed by the application on the screen. This specification [11] defines an API to support these functionalities. This specification assumes the device hosting the TEE to have at least one display screen and one keyboard which must be integrated with device and not physically connected. Remote peripherals are not considered in this specification. The API has three main objectives:

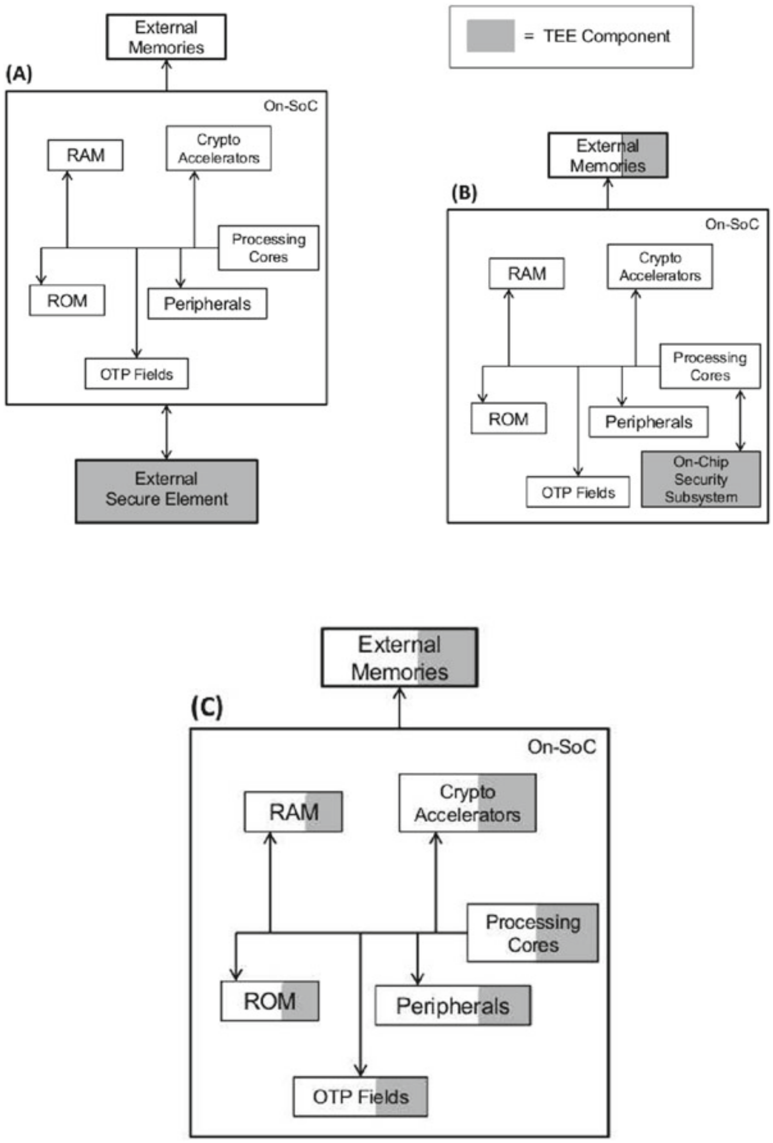


Fig. 18.3 Example realisations of TEE based on [8]

- Secure display: An unauthorised application cannot actively or passively have access to the information displayed by the secure display.
- Secure Input: Any information entered by a user through the keyboard cannot be accessed or modified by any unauthorised application, including trusted applications.
- Secure Indicator: Equally essential, the user should have assurances that the secure display is actually being displayed by an authorised trusted application and therefore can be trusted by the user.

The specification further defines other semantics such as the format of images, number of input fields, formats, and screen orientation.

18.4 Example Implementation of TEEs

There are a number of commercial and open-source implementations of TEEs. In this section, we focus on the most widely deployed implementations.

18.4.1 ARM's TrustZone

ARM's TrustZone is a hardware security technology, and by extension, a system wide security solution is provided by the more recent ARM processors. TrustZone achieves security by logically separating hardware and software resources into two modes, referred to by ARM as the "secure mode"and "normal/non-secure mode". The resources in the secure mode cannot be accessed by the resources in the normal mode.

18.4.1.1 Hardware Architecture

The TrustZone secure/non-secure separation extends to the hardware components, and this is enforced by a hardware logic in the TrustZone-enabled AMBA3 AXI bus fabric [3]. The AMBA3 AXI bus provides an extra *control signal*, the *NS bit*, to read and write channels which are on the main system bus. If NS-bit = 0, then processor is in the secure mode; otherwise, NS-bit = 1 indicates that processor is operating in the non-secure mode. This transition between the two modes is controlled by a mechanism referred to as the *monitor mode* as shown in Fig. 18.4.

TrustZone also secures peripherals such as interrupt controllers, timers, and I/O components through the AMBA3 APB peripheral bus. Securing the interrupt controller ensures that the system is monitored by a task that cannot be interrupted by malware, for example. The secure I/O components such as a "secure keyboard" provide a "trusted path". The AMBA3 APB peripheral bus is a low gate count and

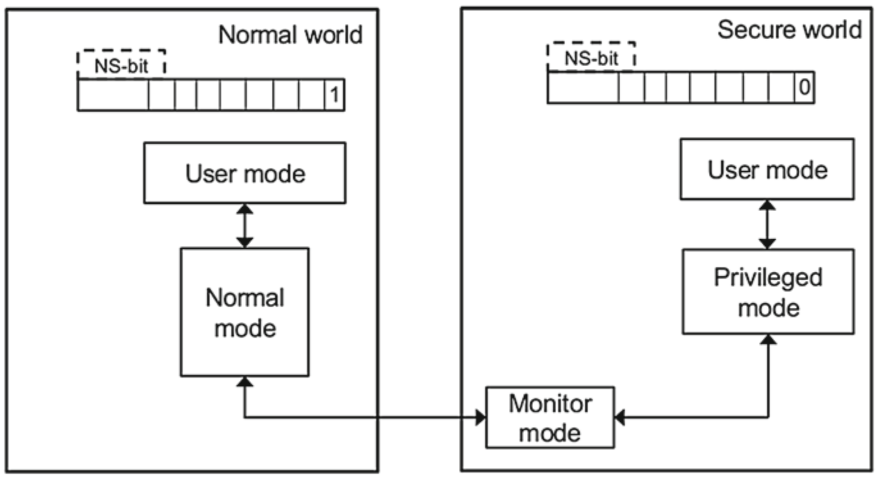


Fig. 18.4 System Architecture of TrustZone: showing the two worlds [3]

low-bandwidth peripheral bus that is connected to the system bus using an AXI to APB bridge. This bridge controls access to the peripherals, ensuring only requests with the necessary security status reach the peripherals.

18.4.1.2 Software Architecture

The exact software architecture for the TrustZone-enabled processor core is implementation dependent. The implementation could be a dedicated "secure-mode OS" or it could just be a software library residing within the secure mode, thereby providing "secure services" to applications in the normal (REE) world. Both implementations are explored below.

- Secure Operating System:
  Although more robust, having a dedicated OS for the secure mode can be regarded as the more intricate of the two options. It allows for the concurrent execution of multiple secure-mode applications independent of the REE. The REE applications can only interact with hardware components that are permitted in the non-secure mode, although they may have access to secure-mode services via secure application drivers. In use, the drivers will cause a temporary switch to the secure mode. It is also possible and perhaps advisable to segregate the secure memory into separate "sandboxes" through the processor's Memory Management Unit (MMU), thereby allowing different trusted applications to access their memory without the need to trust each other.
- Synchronous Library:
  In instances where the complexity of having a dedicated operating system for the secure mode is not essential, a software library running in the secure mode is an

alternative. One of the ramifications of using such an architecture is that it can only handle one task at a time. The library is managed from the non-secure REE and accessed through software calls. It is important to note that in this particular setting, the secure mode cannot run independently as is the case of having a dedicated secure OS.

18.4.1.3 Context Switching

The processor switches between the two modes (context switching) in a *time-sliced* fashion. Time slice is the amount of time a process is allowed to run before it is interrupted and another process is allowed to run. Context switching here is controlled by the “monitor mode”. The monitor mode ensures that transitions from secure mode to non-secure mode and vice versa are smooth. For an application running in non-secure mode to make use of services offered by another application in the secure mode, for example to encrypt a document, a special instruction known as the “Secure Monitor Call (SMC)” is used. SMC immediately sets the NS-bit to “0” and then fully transfers control to the secure mode. On the other hand, switching back from secure mode to non-secure mode is less controlled, and the secure side can directly alter the “current processor status register”. So essentially, the monitor mode can be seen as providing *gatekeeping* services between the two modes.

It is also important to note that the Memory Management Unit (MMU), which translates virtual memory addresses to actual physical memory addresses, has access to the value of the NS-bit flag, thus making it “TrustZone aware”. This means the MMU is able to enforce access control policies on memory locations depending on the particular mode the processor is running in, thereby providing isolation. More technical details on TrustZone can be found in [3].

18.4.2 Samsung KNOX

KNOX is Samsung’s platform security solution for Samsung devices. KNOX realises platform security by relying on a hardware root of trust. It uses an asymmetric key, known as the Device Root Key (DRK), that is uniquely provisioned to each device during its manufacture. The DRK can only be accessed by specially privileged components within the trust zone. And because the DRK is unique to every device, the DRK, or other keys derived from it, can be used to provide other security services such as encryption, attestation, or storage sealing. Data used by the KNOX workspace, for example, is encrypted by this key, thereby providing data protection.

KNOX also provides rollback protection. Rollback protection ensures that a minimum version of software is loaded, to avoid running older and potentially vulnerable versions. KNOX uses hardware fuses encoded with the minimum acceptable version, which is set at the manufacturing stage. The architecture of Samsung KNOX is shown in Fig. 18.5.

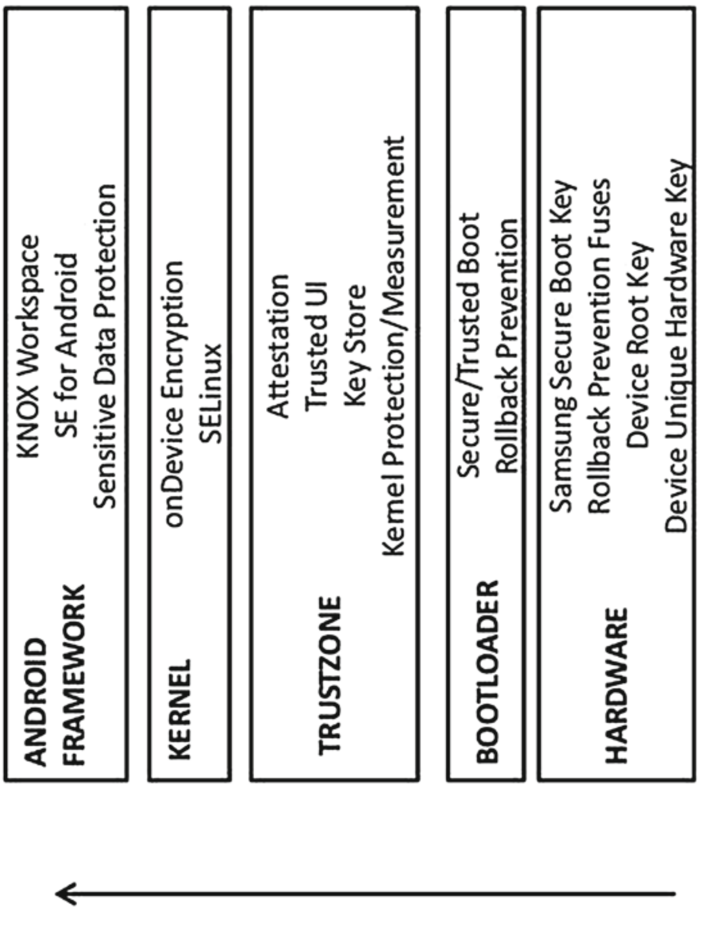


Fig. 18.5 System architecture of Samsung KNOX

Typically, the Android boot process begins by starting up from ROM, with the first stage of the boot process known as the “primary boot loader”. The boot loader does some system initialisations and loads the secondary boot loader into RAM for execution. Depending on specific implementations, there could be one or more secondary boot loaders with each boot loader loading the next in a sequential and predetermined manner. The final boot loader is the Android boot process, known as aboot, which starts the Android operating system. To ensure only known, trusted, and authorised boot loaders are loaded, KNOX provides both secure and trusted boot. In secure boot, each boot loader verifies the integrity of the next stage in the boot process, using a digital signature with verification keys stored in hardware. If this verification fails at any time, the whole process is terminated. However, it is possible to have two versions of the same software, both with a valid signature, for example a version of firmware is found to be vulnerable, and subsequently patched. Both versions will have a valid signature and secure boot is not able to differentiate between the two. Furthermore, a provider might want to allow custom kernels to run on their devices. In cases like this, secure boot is not enough. A mechanism is needed that lets “anything” load, but a cryptographic measurement, typically in the form of a hash must be taken and reliably stored and reported. KNOX relies on TrustZone to do that. Later when the device is operating, applications and services can make reference to the measurement before taking decisions.

18.4.3 Intel Software Guard Extensions (SGX)

Intel SGX provides a set of instructions that provide extensions to the Intel architecture processors. The goal of these extensions is to provide trusted execution environment within the computer, where applications including privileged software cannot be trusted. Intel SGX thus aims to provide confidentiality, integrity, and replay protection using *enclaves* [12].

As shown in Fig. 18.6, an enclave is a region within the applications’ memory space which is hardware protected. An enclave has a reserved area of memory from which it runs, known as the Enclave Page Cache (EPC). Access to the EPC is protected from processes outside the enclave. The OS can only manage the enclaves but cannot tamper with the code and data within the enclave itself. This reduces the TCB and thus the attack surface, because an application does not have to explicitly trust the OS running on top of the hardware.

18.4.3.1 Enclave Life Cycle

The management of an enclave is hardware-backed for enhanced security and is carried out through a set of instructions. An enclave is created using the *ECREATE* instruction. This creates the enclave and also sets the base linear address as well as the physical address. After the creation of an enclave, the *EADD* instruction is used

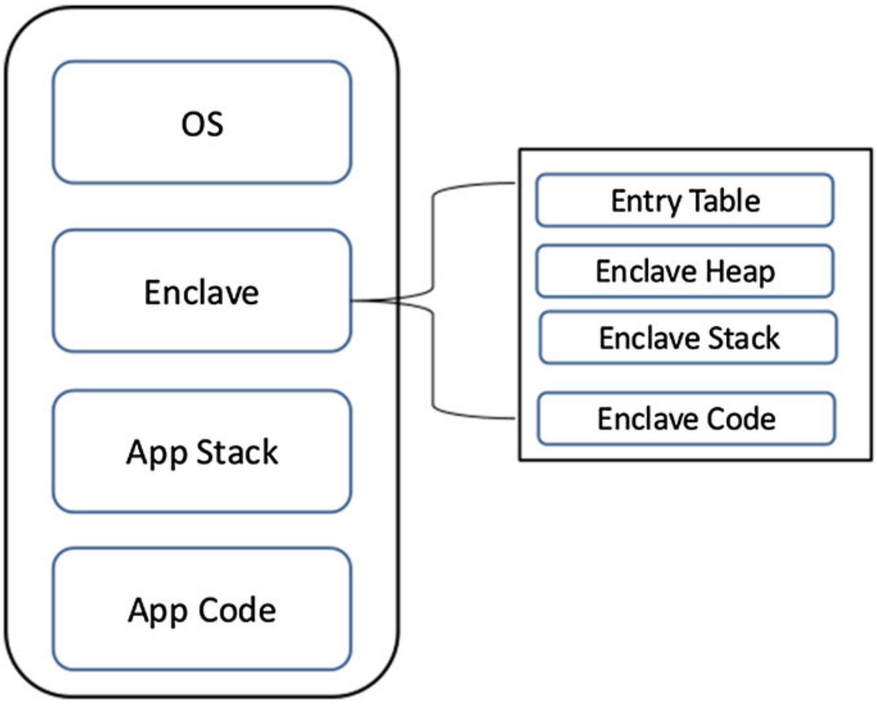


Fig. 18.6 Intel SGX enclave within the application's address space based on [12]

to add relevant code and data to the enclave. This adds 4KB of protected data. For integrity protection, the *EEXTEND* instruction is used to measure the contents of the enclave. This measures 256 bytes at a time; therefore, to measure the whole contents of the enclave, the instruction is called 16 times [12]. The enclave is then initialised using the *EINIT* instruction. This sets the INIT attribute to true, which means the code in the enclave can be executed (Fig. 18.7).

Access to the enclave or its data can be direct or indirect. Direct enclave access comprises of all memory access originating from the enclave itself while indirect accesses are those originating from the SGX instructions. These instructions are used as control messages and include instructions such as ECREATE and EADD.

18.5 Host-Based Card Emulation

Near-Field Communication (NFC) in card emulation mode permits a mobile device to emulate a contactless smart card and exchange Application Protocol Data Units (APDU)[2] with an external card reader. From the card reader's point of view, the

[2] APDU is a unit of communication between a smart card and a reader.

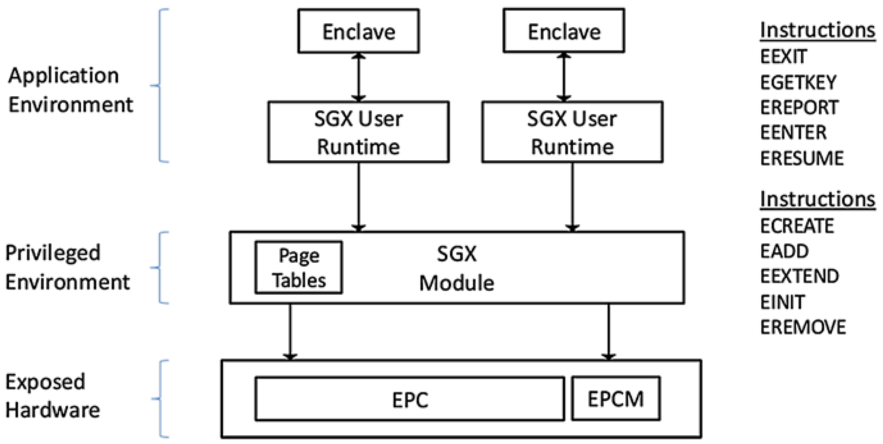


Fig. 18.7 Hardware and software architecture of Intel SGX based on [12]

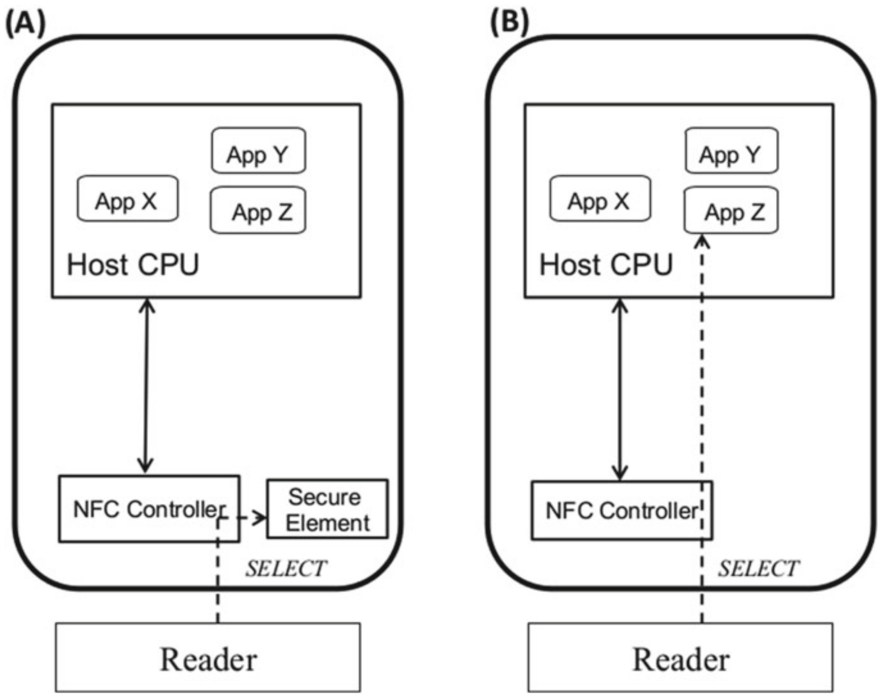


Fig. 18.8 Diagrams showing NFC SE-based card emulation and Host Card Emulation

mobile device appears and behaves like any other smart card. Traditionally, NFC card emulation mode has relied on a tamper-resistant hardware Secure Element (SE) to provide security protection. This is because the host OS cannot provide the level of application security required by some service providers. The SE provides a secure storage and execution environment for the applications emulating the smart card. When an NFC emulation device is tapped on a reader, command APDUs from the reader are received by the NFC controller, via the NFC antenna, and are routed to the SE as shown in (A) part of Fig. 18.8.

However, the use of an SE introduces practical constraints for an application developer and/or service provider. Access to the SE is usually tightly controlled, requiring complex business agreements before an application, or related data is provisioned to the SE.

Host Card Emulation (HCE) [13], on the other hand, is a means for an application running on the host OS of the device to emulate a smart card. This means that the NFC controller is effectively able to route APDUs to the application directly, bypassing the SE. This is depicted in Fig. 18.8. Therefore, in HCE, security (attack protection) is traded to facilitate NFC service development and roll-out.

The notion of HCE (or its equivalent) was first evident on a mobile device in the Blackberry 7 OS and was referred to as “virtual target emulation”. In 2012, SimplyTapp[3] came up with HCE in an effort to break the reliance on the SE and thereby make NFC card emulation more accessible. SimplyTapp added patches to the CyanogenMod [14] OS. More notably, Google included support for HCE in Android, starting from Android 4.4 (KitKat).

18.5.1 Functional Differences Between SE-Based Card Emulation and HCE

While both methods let an NFC device emulate a smart card, they are fundamental differences from a functional point of view. This subsection highlights these differences.

- **Power Mode:**
  SE-based card emulation has significant advantages over HCE in that it can operate in *low power mode*. This means even when the device’s battery is in *battery low mode*.[4] The power generated by the external NFC reader is sufficient to power the NFC controller for transactions. HCE on the other hand does not support the low power mode because the whole operating system has to be powered on, for the card emulating application to function. This is important in scenarios such as transport ticketing, where the user is typically expected to tap the device at points of *entry* and *exit*, and drained batteries are common.

[3]A startup in the USA, https://www.simplytapp.com/.

[4]Battery low in this context means the device’s battery is too low to power on the OS and UI for regular usage. And “battery off” means the battery is either not present, or it has no residual power to even support the NFC controller.

- **Transaction Speed:**
  HCE-based applications have proven to produce higher transaction speeds than similar SE-based implementations [15]. This is because HCE applications can leverage more computing resources provided by the host CPU, as compared to the limited computing resources of the SE. Nevertheless, HCE-based applications are subject to higher variations in terms of their performance compared to SE-based implementations. This is because HCE applications compete with many more processes concurrently running on the host CPU. This means their performance largely depends on the CPU load at the time of the transaction, which is not the case with SE-based applications. These variations have both functional and security implications. More details can be found in [15].
- **Data Connectivity:**
  Although pure HCE can function without data connectivity, the need to mitigate the risk of exposure means it will require certain measures to be in place. One of the most feasible ways is to transfer all sensitive material to the cloud and provision them at the time of transaction. However, there are use cases such as transport ticketing, where connectivity cannot be guaranteed. In such cases, the usability of HCE with cloud-based storage may be called into question.

18.5.2 Security Considerations

The reliance on software in HCE-based transactions introduces certain risks. For example, malware on the device may be able to monitor transactions and steal sensitive information such as payment credentials. A number of measures have been considered to mitigate this inherent risk of exposure introduced by HCE, which are explained below. However, none of these measures provide the same levels of security as a hardware SE.

- **Cloud-based Storage:**
  The idea here is to store all sensitive credentials in the cloud and then only provision them to the device when needed i.e. just before a transaction. To use HCE with cloud-based storage, some issues need to be further addressed. For example, the device must be authenticated prior to granting a provisioning request. The integrity of the mobile application must be verified and able to be attested by the cloud service. Cloud-based storage however has its own disadvantages, notably that the Internet connection between the mobile device and the cloud must be maintained at transaction time. This connection introduces latency which may be unacceptable in some use cases such as transit and payment.
- **Tokenisation:**
  This is the process of replacing sensitive data such as the Primary Account Number (PAN), with a surrogate value (token) that has a shorter temporal validity than the original. In the case of a compromise, this ensures that the token is of limited use to the attacker.

Although tokenisation itself is not a new concept, recent developments in mobile payments have led to renewed interest. Notably, EMVCo[5] has released specifications [2] on the usage of tokenisation for mobile payments.

- **White-Box Cryptography:**
  First published in [16], white-box cryptography is a software implementation of a cryptographic algorithm that can resist *white-box attacks* [17]. White-box attacks, in this instance, assume that the attacker can gain full control of the target. This includes attaining the highest privileges on the target, full access to the algorithms implementation details, as well as being able to observe sensitive data during dynamic execution. Therefore, traditional cryptographic algorithm implementations of this form of software protection obfuscate cryptographic keys by embedding them into the cryptographic algorithm's code in such a way that an attacker with access to the code will not be able to deduce the keys.
- Trusted Execution Environment (TEE): A TEE can be used to provide a secure execution environment for HCE applications. The TEE can also be used to provide a *trusted UI* for HCE applications. Secure entry of PINs for payment applications is a typical example. This segregation ensures that a HCE-based application in the trusted side of the device is protected against malware, as well as other applications in the rich OS. Another implementation option could be having the HCE application running on the rich OS, while delegating security sensitive operations such as encryption, and random number generation to the TEE.

18.5.3 HCE Application Development: Android Example

Android's HCE architecture is centred around Android services [18] known as "HCE services". Services have behaviours that are suitable for HCE applications for use in areas such as ticketing and payments. A service can run in the background and does not need user interference to operate. This means a user can simply tap the device against a reader for the transaction to go through, without necessarily launching the application.

The implementation of an Android-based HCE application generally involves two stages: application selection and data exchange.

18.5.3.1 Application Selection

In a case where there is more than one HCE application on the device, application selection ensures that the correct application responds when the device is tapped against a reader. All HCE applications running on the Operating System (OS) are identified using the Application Identifier (AID). The AID is registered at the begin-

[5]EMVCo, made up of six members: American Express, Discover, JCB, MasterCard, UnionPay, and Visa, facilitates worldwide interoperability and acceptance of secure payment transactions.

ning of development in the Android manifest file. The first command APDU received by the NFC controller is the *SELECT* command. The selection of applications using the AID is specified in ISO 7816-4 standards and more details can be found in [19]. The SELECT command contains the AID of the application that the external reader wishes to communicate with. The NFC controller looks up the AID in its *routing table* and makes a decision on which application the AID is for. Subsequently, all further APDUs are sent to the corresponding application until a new SELECT command is received or the NFC link is broken [13].

18.5.4 Data Exchange

Card emulation using HCE requires the application to have a *service* component that handles NFC transactions. All services *extend* the *HostApduService*. The HostApduService declares two abstract methods that as a minimum are overridden and implemented. (Example code is shown in Listing 18.1.)

```
public class MyHostApduService extends HostApduService {
    @Override
    public byte[] processCommandApdu(byte[] apdu, Bundle extras) {
        ...
    }
    @Override
    public void onDeactivated(int reason) {
        ...
    }
}
```

Listing 18.1 Data Exchange

The *processCommandApdu()* method is called whenever the service receives a command APDU from an NFC reader. Response bytes should be sent back almost immediately for a seamless user experience because the user will most likely be holding the device against the NFC reader [20]. Otherwise, the response can be sent later using the *sendResponseApdu()* method. The *onDeactivated()* method is called when the NFC link is broken for any reason, or when a new SELECT command with a different AID is received. The *onBind()* method is used to return the communication channel to the service. The *notifyUnhandled()* is used to notify the OS when the service cannot complete the transaction.

18.6 Conclusion

In this chapter, the concept and technologies of TEE have been explored, as a means to provide assurance that critical security-sensitive applications run in protected isolation on mobile devices. TEEs provide platform integrity assurance, as well as secure storage and execution environments, upon which foundations, other security services such as remote attestation and trusted path, can be built. Host Card Emulation as a software alternative to card emulation with a hardware SE has also been discussed. The inherent security risks have also been highlighted, along with suggestions for managing these risks.

At the time of writing[6], there is considerable effort in the TEE space, with developments in standards, such as via the TCG, UEFI, and GlobalPlatform, as well as commercial implementations building upon ARM's TrustZone and Intel's SGX hardware. Much of the drive for this is to protect general and sensitive mobile applications from malware; however, this in turn benefits HCE. Although HCE is unlikely to rival the attack resistance of a dedicated hardware SE, improvements in TEE security may mean that temporary credentials (as in tokenisation) may be used for longer periods.

Some concerns on the horizon include the fact that the TEE implementations are proprietary and normally not openly released for security peer review and that the controls needed to safeguard TEE security, as well as commercial interests, may recreate the barriers that prevented service provider access to the hardware SEs.

References

1. Amit Vasudevan, Emmanuel Owusu, Zongwei Zhou, James Newsome, and Jonathan M. McCune. *Trust and Trustworthy Computing: 5th International Conference, TRUST 2012, Vienna, Austria, June 13-15, 2012. Proceedings*, chapter Trustworthy Execution on Mobile Devices: What Security Properties Can My Mobile Platform Give Me?, pages 159–178. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. Cited 06 Jan 2016.
2. EMV Payment Tokenisation Specification. Standard, 2014. Cited 15 Jan 2016.
3. ARM Limited. ARM Security Technology Building a Secure System using TrustZone Technology, April 2009. http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf. Cited 08 Feb 2016.
4. Trusted Computing Group. TCG Specification TPM 2.0 Mobile Reference Architecture, December 2014. http://www.trustedcomputinggroup.org/wp-content/uploads/TPM-2-0-Mobile-Reference-Architecture-v2-r142-Specification_FINAL2.pdf. Cited 17 Feb 2016.
5. Trusted Computing Group. TCG Specification TPM 2.0 Mobile Common Profile, December 2015. http://www.trustedcomputinggroup.org/wp-content/uploads/TPM_2.0_Mobile_Common_Profile_v2r31.pdf. Cited 19 Feb 2016.
6. Trusted Computing Group. TCG Specification TPM 2.0 Mobile Command Response Buffer Interface, December 2014. http://www.trustedcomputinggroup.org/wp-content/uploads/Mobile-Command-Response-Buffer-Interface-v2-r12-Specification_FINAL2.pdf. Cited 19 Feb 2016.
7. Unified Extensible Firmware Interface Forum. Unified Extensible Firmware Interface Specification–version 2.6, January 2016. http://www.uefi.org/sites/default/files/resources/UEFIUEFI%20Spec%202_6.pdf. Cited 02 Jan 2016.
8. GlobalPlatform. GlobalPlatform Device Technology, TEE System Architecture v1.0, December 2011. Cited 06 Mar 2016.
9. GlobalPlatform. GlobalPlatform Device Technology, TEE Client API Specification v1.0, July 2010. Cited 06 Mar 2016.

[6] August 2016.

10. GlobalPlatform. GlobalPlatform Device Technology, TEE Internal API Specification, December 2011. Cited 10 Mar 2016.
11. GlobalPlatform. GlobalPlatform Device Technology, Trusted User Interface API Specification v1.0, June 2013. Cited 12 Mar 2016.
12. Intel Corporation. Intel Software Guard Extensions Programming Reference, October 2014. https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf. Cited 18 Mar 2016.
13. Smart Card Alliance. Host Card Emulation (HCE) 101. Technical report, Smart Card Alliance, Mobile and NFC Council, August 2014. Cited 20 Mar 2016.
14. Doug Yeager. Added NFC Reader support for two new tag types: ISO PCD type A and ISO PCD type B, 2012. https://github.com/CyanogenMod/android_packages_apps_Nfc. Cited 06 Apr 2016.
15. Assad Umar, Keith Mayes, and Konstantinos Markantonakis. Performance variation in host-based card emulation compared to a hardware security element. In *First Conference on Mobile and Secure Services (MOBISECSERV)*, pages 1–6, 2015. Cited 11 Apr 2016.
16. Stanley Chow, Phil Eisen, Harold Johnson, and Paul C. van Oorschot. *Digital Rights Management: ACM CCS-9 Workshop, DRM 2002, Washington, DC, USA, November 18, 2002. Revised Papers*, chapter A White-Box DES Implementation for DRM Applications, pages 1–15. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. Cited 16 Apr 2016.
17. Brecht Wyseur. White-box cryptography: Hiding keys in software. Technical report, NAGRA Kudelski Group, Switzerland, 2012. Cited 06 Apr 2016.
18. Android Developer Guide. Service. https://developer.android.com/reference/android/app/Service.html#WhatIsAService. Cited 16 Apr 2016.
19. Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange. Standard, International Organization for Standardization, Geneva, CH, 2013. Cited 06 Jun 2016.
20. Android Developer Guide. Host-based card emulation. https://developer.android.com/guide/topics/connectivity/nfc/hce.html. 16 Apr 2016

Index