Michael Hutter
Jörn-Marc Schmidt (Eds.)

LNCS 8262

Radio Frequency Identification

Security and Privacy Issues

9th International Workshop, RFIDsec 2013
Graz, Austria, July 2013
Revised Selected Papers

Springer

Michael Hutter · Jörn-Marc Schmidt (Eds.)

Radio Frequency Identification

Security and Privacy Issues

9th International Workshop, RFIDsec 2013
Graz, Austria, July 9–11, 2013
Revised Selected Papers

Springer

Editors
Michael Hutter
Jörn-Marc Schmidt
Graz University of Technology
Austria

ISSN 0302-9743 ISSN 1611-3349 (electronic)
ISBN 978-3-642-41331-5 ISBN 978-3-642-41332-2 (eBook)
DOI 10.1007/978-3-642-41332-2
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2013953279

CR Subject Classification K.6.5, E.3, K.4.4, C.2, C.3

Preface

RFIDsec 2013, the 9th workshop on RFID Security and Privacy, was held in Graz, Austria, during July 9–11, 2013. More than 50 participants from 13 countries attended the workshop. The Program Committee consisting of 24 experts in the field selected 11 papers out of 23 submissions for publication in the workshop proceedings. Each paper was reviewed by at least three and on average four reviewers.

The program included three invited talks given by international experts in the field. The first invited talk entitled "RFID Privacy: From Transportation Payments to Implantable Medical Devices" was given by Wayne Burleson, who introduced the audience to state of the art RFID privacy issues. The second talk was given by Günther Lackner and Karin Greimel, who talked about "20 Years of MIFARE. From CRYPTO1 to Formal Verification." They gave an overview of various industry perspectives in the case of the Mifare product family that celebrates its 20 years anniversary. The third talk was given by Lejla Batina entitled "How Light Is Lightweight Crypto". Her talk focused on RFID identification protocols and hardware-implementation requirements for RFID.

Since 2011, RFIDsec has additionally provided area-relevant tutorials for attendees from academia and industry. In this year's edition, we offered three tutorials that preceded the workshop. The specific topics were "RFID Introduction and the IAIK DemoTag: A Programmable RFID-Tag Emulator" given by Thomas Korak, Raphael Spreitzer, and Hannes Gross, "Side-Channel Attacks and Fault Analysis" given by Johann Heyszl and Thomas Korak, and "Cryptographic Hardware Design and Performance Metrics" given by Frank K. Gürkaynak.

This year's edition of RFIDsec was the first in cooperation with the International Association for Cryptologic Research. We would like to thank Bart Preneel for promoting and supporting us to get the IACR "In Cooperation" status. Furthermore, we would like to thank the sponsorships from the Styrian Business Promotion Agency (SFG), the City of Graz, the State of Styria, and NXP Semiconductors. With their support it was possible to provide student stipends for attending the workshop.

Finally, we would like to thank all authors for their paper submission, the Program Committee and all external reviewers for their support in the review process allowing us to accept a number of high-quality papers, and all the attendees of the workshop for their active participation and contribution to this research area.

August 2013

Michael Hutter
Jörn-Marc Schmidt

RFID: Contactless Identification and Security Technology

Radio-frequency identification (RFID)[1] is a technical system that offers the possibility of reading data through radio waves without the need for contact. This allows the automatic identification and location of objects and makes the collection of data easier.

The most exciting thing about RFID is its diverse application range, which allows both security and comfort for the end user. From passports to logistic processes and patient follow-up, from time recording to bus tickets and engine immobilizers, the application possibilities seem endless and the market potential is huge. And Styria is involved in a big way!

RFID-Hotspot Styria[2]: Styria, the second largest province of Austria, has about 1.2 million inhabitants and is situated central to the emerging markets of south-eastern Europe with about 20 million people. Graz, the capital city of Styria, is an old university town and has a population in the greater area of about 400,000 long-term inhabitants. With seven universities, a broad range of R&D and competence centers, and its own research institution called Joanneum Research, Styria is Austria's top engineering, science, and research province.

The development of the RFID technology has a long tradition in Styria. Styrian RFID companies — and in Styria there are a collection of world-leading companies (NXP, Infineon, ams etc.) — are highly renowned in this field: More than 50% of RFID chips in use worldwide have been developed in Styria. Practically all of these companies are operational at an international and worldwide level and at present employ ca. 2,000 people (primarily in Graz and its surrounding area). The export ratio in the RFID sector in Styria stands at over 90%. Global brands such as Mifare, Hitag, Legic were developed by Styrian companies and have gone on to become world-leading brands. Further data and facts on Styria as an RFID hotspot are as follows:

- Styria offers expertise along the entire value chain (incl. research and education): Chip, Antennas, Reader, Software, System provider.
- More than 50% of RFID chips currently in global use have been developed in Styria.
- Styria is the birthplace of Near Field Communication (NFC).

[1] RFID is part of the core competency "Electronics, Instrumentation and Control Technology" of the Economic Strategy Styria 2020 (http://sfg.at/cms/3724/).

[2] The RFID-Hotspot Styria was initiated by the Styrian Business Promotion Agency (SFG). SFG is a service provider, which aims to contribute to the consolidation and growth of the Styrian economy. As a 100% subsidiary of the Styrian Government, SFG operates all business support tasks for its owner. This is the provision of monetary support with a broad range of grant and financing programs, as well as tasks like raising and steering clusters and networks, technology parks, technology transfer, and the consulting of foreign investors. For more information about the RFID-Hotspot Styria, please visit our website (http://sfg.at/rfid) or contact us via e-mail (rfid@sfg.at).

– With a market share of over 95% in the field of RFID and security, chips originating from Styria for application in, e.g., government documents (passports, driver's licences etc.) are in use in over 100 countries.
– The no. 1 innovation used in readers for passport chips comes from Styria.
– In the RFID automotive sector (engine immobilizers, radio controlled keys, tyre pressure sensors etc.), chip innovations from Styria are also no. 1 with a market share of 50%.
– Styria is no. 1 in the area of chip innovation with well-known applications such as access control, electric ticketing, and electronic payments.

What would future technology be without its corresponding research activities? In Styria there are an array of universities and other research and development institutes that are active in the area of RFID. Universities, polytechnics, and non-university research institutes do not just provide us with research results but also with the raw material for future business location development: qualified employees. With its renowned RFID companies and accompanying research and educational activities, Styria is a Mecca for RFID where future RFID developments are forged.

The following research and educational initiatives are examples of the cooperation between economy, science, and education in Styria in the field of RFID:

SeCoS — Secure Contactless Sphere: Representatives from the entire RFID supply chain have come together to form a consortium to work on the K-Project SeCoS (lead partner: Joanneum Research). The aims of the project are to develop a platform that places the highest demands on security and protection of privacy all the way from the chip to the application itself as well as to reduce component size, to enhance carrier frequencies and data transfer rates, and to improve the precision of object tracking. Several application scenarios will be implemented to demonstrate the research results. The focus on security and privacy technology originates from the fact that new applications that involve RFID regularly raise concerns regarding these topics. This is because RFID tags can be read out over a distance without any interaction of the user; the user is not even aware that the tag is/was read out. Within the project, we want to provide a comprehensive methodology for making use of the advance of RFID tags yet ensuring privacy of all parties involved in the product life-cycle.

This K-Project SeCoS is funded in the context of COMET - Competence Centers for Excellent Technologies by BMVIT, BMWFJ, SFG, Province of Styria, Government of Styria. The program COMET is conducted by the Austrian Research Promotion Agency (FFG).

RFID Qualification Network Austria: Based on a study conducted by the SFG, TU Graz and various partners from science and business make up the RFID Qualification Network Austria, which is supported by the Austrian Research Promotion Agency (FFG). The Network has succeeded in establishing a consortium of 24 businesses and educational institutions, covering the entire value-added chain of the RFID area. February 2013 saw the start of a comprehensive continuing education program comprising 48 individual courses offered by TU Graz and FH Campus 02. In terms of content, focuses include state-of-the-art technological developments, the building of competences in innovation and demand, the integration of RFID into businesses, and know-how on key factors of RFID systems.

Both science and businesses benefit from this cooperation in many ways. The intensive dialogue between participants and instructors influences new innovation and research projects, for example, in the areas of software architecture, Web services, IT security and tool-supported software development. In addition to Graz University of Technology as the applicant and many small and medium-sized businesses, FH Campus 02, Joanneum Research, AVL List, ams, NXP, Infineon, Voest Alpine, and the Evolaris Competence Centre are on board as well.

The goal for the near future is to establish Graz as a leading international educational region for RFID by providing post-graduate university courses, study programs, and courses that go beyond the training currently on offer.

Organization

RFIDsec 2013 was organized by the Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Austria.

Executive Committee

Program Co-chairs

Michael Hutter	TU Graz, Austria
Jörn-Marc Schmidt	TU Graz, Austria

General Chair

Jörn-Marc Schmidt	TU Graz, Austria

Program Committee

Gildas Avoine	Université Catholique de Louvain (Belgium)
Lejla Batina	Radboud University Nijmegen (The Netherlands)
Mike Burmester	Florida State University (USA)
Srdjan Capkun	ETH Zurich (Switzerland)
Paolo D'Arco	University of Salerno (Italy)
Thomas Eisenbarth	Worcester Polytechnic Institute (USA)
Martin Feldhofer	NXP Semiconductors (Austria)
Julio Hernandez-Castro	University of Kent (UK)
Jaap-Henk Hoepman	TNO / Radboud University Nijmegen (The Netherlands)
Timo Kasper	Ruhr University Bochum (Germany)
Kerstin Lemke-Rust	Hochschule Bonn-Rhein-Sieg (Germany)
Yingjiu Li	Singapore Management University (Singapore)
Nai-Wei Lo	National Taiwan University of Science and Technology (Taiwan)
Konstantinos Markantonakis	Royal Holloway University of London (UK)
Aikaterini Mitrokotsa	École Polytechnique Fédérale de Lausanne (Switzerland)
Karsten Nohl	Security Research Labs (Germany)
Berna Örs	Istanbul Technical University (Turkey)
Pedro Peris-Lopez	Carlos III University of Madrid (Spain)

Axel Poschmann	Nanyang Technological University (Singapore)
Pankaj Rohatgi	Cryptography Research Inc. (USA)
Kazuo Sakiyama	University of Electro-Communications (Japan)
Nitesh Saxena	University of Alabama at Birmingham (USA)
Marc Witteman	Riscure (The Netherlands)
Avishai Wool	Tel Aviv University (Israel)

External Reviewers

Xavier Carpent
Baris Ege
Jian Guo
Yoshikazu Hanatani
Gesine Hinterwalder
Divyan Konidala
Anna Krasnova
Yang Li
Luka Malisa
Tania Martin
Ramya Masti
Shugo Mikami
Manar Mohamed
Phuong Ha Nguyen
David Oswald
Babins Shrestha
Xin Ye

Sponsoring Institutions

Styrian Business Promotion Agency (SFG)
City of Graz
State of Styria
NXP Semiconductors

Contents

NFC and Mobile Security

Deploying OSK on Low-Resource Mobile Devices 3
Gildas Avoine, Muhammed Ali Bingöl, Xavier Carpent, and Süleyman Kardaş

Is NFC a Better Option Instead of EPC Gen-2 in Safe Medication of Inpatients 19
Mehmet Hilal Özcanhan, Gökhan Dalkılıç, and Semih Utku

Rights Management with NFC Smartphones and Electronic ID Cards: A Proof of Concept for Modern Car Sharing 34
Timo Kasper, Alexander Kühn, David Oswald, Christian Zenger, and Christof Paar

Protocols and Attacks

Desynchronization and Traceability Attacks on RIPTA-DA Protocol 57
Nasour Bagheri, Praveen Gauravaram, Masoumeh Safkhani, and Somitra Kumar Sanadhya

Long Distance Relay Attack 69
Luigi Sportiello and Andrea Ciardulli

On the Security of Two RFID Mutual Authentication Protocols 86
Seyed Farhad Aghili, Nasour Bagheri, Praveen Gauravaram, Masoumeh Safkhani, and Somitra Kumar Sanadhya

RFID Hardware

Dietary Recommendations for Lightweight Block Ciphers: Power, Energy and Area Analysis of Recently Developed Architectures 103
Lejla Batina, Amitabh Das, Barış Ege, Elif Bilge Kavun, Nele Mentens, Christof Paar, Ingrid Verbauwhede, and Tolga Yalçın

An Improved Hardware Implementation of the Quark Hash Function 113
Shohreh Sharif Mansouri and Elena Dubrova

Analyzing Side-Channel Leakage of RFID-Suitable Lightweight ECC Hardware 128
Erich Wenger, Thomas Korak, and Mario Kirschbaum

Implementations

Energy-Architecture Tuning for ECC-Based RFID Tags 147
Deepak Mane and Patrick Schaumont

Speed and Size-Optimized Implementations of the PRESENT Cipher for Tiny AVR Devices 161
Konstantinos Papagiannopoulos and Aram Verstegen

Author Index 177

NFC and Mobile Security

Deploying OSK on Low-Resource Mobile Devices

Gildas Avoine[1], Muhammed Ali Bingöl[2,3](✉), Xavier Carpent[1], and Süleyman Kardaş[2,3]

[1] ICTEAM Institute, Université Catholique de Louvain, 1348 Louvain la Neuve, Belgium
[2] TÜBİTAK BİLGEM UEKAE, Gebze, Kocaeli, Turkey
muhammedali.bingol@tubitak.gov.tr
[3] Faculty of Engineering and Natural Sciences, Sabancı University, 34956 İstanbul, Turkey

Abstract. It is a popular challenge to design authentication protocols that are both privacy-friendly and scalable. A large body of literature in RFID is dedicated to that goal, and many inventive mechanisms have been suggested to achieve it. However, to the best of our knowledge, none of these protocols have been tested so far in practical scenarios. In this paper, we present an implementation of the OSK protocol, a scalable and privacy-friendly authentication protocol, using a variant by Avoine and Oechslin that accommodates it to time-memory trade-offs. We show that the OSK protocol is suited to certain real-life scenarios, in particular when the authentication is performed by low-resource mobile devices. The implementation, done on an NFC-compliant cellphone and a ZC7.5 contactless tag, demonstrates the practicability and efficiency of the OSK protocol and illustrates that privacy-by-design is achievable in constrained environments.

Keywords: RFID authentication · Implementation · Time-memory trade offs · Privacy

1 Introduction

A major research topic in RFID is the development of authentication protocols that respect the privacy of the users, while still being efficient enough to be applicable in large-scale systems. When the time needed by the authentication process is not negligible, the user must hold the card steady in front of the reader until reception of an audio or visual signal. Long authentication processes are not practical in access control systems where delaying the customer flow is not acceptable for example in mass transportation or cultural events. It is generally agreed upon that approximately 200 ms can be dedicated to grant or deny the access to a customer in a flow [10].

Classical challenge-response protocols such as ISO/IEC 9798-2.2 [14] can be privacy-friendly if the tag (prover) does not send its identifier in the clear to the reader (verifier). In such a case, the reader must find the tag identifier by performing an exhaustive search in its database. For example, in a system managing 2^{20} tags and a reader capable of performing 2^{20} cryptographic operations per second, the authentication of a tag takes half a second on average, which is beyond the time threshold that can be allocated to the security operations.

Several protocols have been designed to provide privacy in the authentication. An additional property, named forward privacy, ensures that if a tag is compromised at one point, an adversary is not able to trace it in the past (given past communication traces). An example of such a protocol is the OSK protocol, proposed by Ohkubo, Suzuki and Kinoshita in [20]. It may be regarded as one of the most privacy-friendly protocol among the ones that allow for an efficient authentication procedure based on symmetric-key cryptography [1].

An implementation of this protocol was previously done in [6]. It uses rainbow tables accommodated for OSK, as proposed by Avoine and Oechslin in [5]. However, we show in this paper that in the setting of [6], a faster and simpler approach is viable, namely the full storage. Instead, we focus on systems with low-resource mobile readers, such as PDA's or NFC-compliant cellphones, and adapt this implementation to that context. We show that such a protocol with very good privacy properties is efficient enough to be used in practice, even in such constrained environments.

The structure of the paper is as follows. We present the OSK protocol in Sect. 2 and the adapted time-memory trade-off in Sect. 3. The method of Avoine and Oechslin for accommodating OSK for time-memory trade-offs is described in Sect. 4. We describe our implementation and discuss our results in Sect. 5. We finally conclude in Sect. 6.

2 Ohkubo, Suzuki, and Kinoshita's Protocol

2.1 Description

The OSK protocol is proposed by Ohkubo, Suzuki, and Kinoshita in [20]. It is one of the most well-known RFID-devoted authentication protocols and is the earliest one that achieves *forward privacy.*[1] In the RFID context, forward privacy is the property that guarantees the security of past interactions of a tag even if it is compromised at a later stage. Namely, the secret information of a tag $\mathcal{T}_i (1 \leq i \leq n)$ is corrupted by an adversary at time t, the adversary can not associate any transaction with $\mathcal{T}_i$ at any time $t' < t$.

In the OSK protocol, each tag $\mathcal{T}_i$ has an initial secret S_i^0 that is updated after each authentication query. The update consists in hashing the current secret with the one-way function $\mathcal{H}$. Upon reception of the authentication query, the tag answers by hashing the current secret with a different[2] hash function, $\mathcal{G}$. Figure 1 shows the OSK protocol.

[1] It is also known as *backward untraceability* and used interchangeably in some papers [16,18,21].

[2] Note that although these two functions need to be different, only one algorithm may be implemented on the tag, and an additional 1-bit input parameter used to select the function.

$$\begin{array}{ccc}
\mathcal{R} & & \mathcal{T}_i \\
 & \xrightarrow{\text{request}} & \\
\text{find } S_i^0 \text{ s.t.} & \xleftarrow{\sigma=\mathcal{G}(S_i^j)} & S_i^{j+1} = \mathcal{H}(S_i^j) \\
\exists j \geq 0 : \ \mathcal{G}(\mathcal{H}^j(S_i^0)) = \sigma & &
\end{array}$$

Fig. 1. OSK protocol

System Setup. Each tag $\mathcal{T}_i$ of the system is initialized with a randomly chosen secret S_i^0. The n initial secrets are stored in a database, sometimes called back-end system. In some settings the back-end system and the reader are two different devices, connected in a way that is considered secure. In some other settings the back-end system is embedded in the readers.

Interrogation. When the tag is queried by a reader it answers with a response using the current secret such that $\sigma = \mathcal{G}(S_i^j)$ and also updates the secret immediately using a different hash function: $S_i^{j+1} = \mathcal{H}(S_i^j)$.

Search and Identification. When receiving an answer the database searches for an initial secret S_i^0 that leads to σ. In other words, it checks whether there exists i and j such that $\mathcal{G}(\mathcal{H}^j(S_i^0)) = \sigma$. To do that, from each of the n initial secrets S_i^0, the reader computes the hash chains as shown in Fig. 2 until it finds a value matching σ, or until it reaches a given maximum limit L (the "lifetime" of a tag) on the chain length.

The value $\sigma = \mathcal{G}(S_i^j)$ does not leak any information to an attacker on the secret of $\mathcal{T}_i$ when $\mathcal{G}$ and $\mathcal{H}$ behave as pseudo-random functions. However, given that the authentication process is bounded by L, the OSK protocol is prone to desynchronization when an adversary queries the tag more than L times. In such a case, the tag can no longer be authenticated and a privacy issue arises for a certain type of adversary, capable of detecting the success status of an authentication (see [15] for a discussion). Fortunately, the synchronization can be retrieved by the back-end without exchanging the tag; for example, the holder of a *desynchronized* tag can ask the system operator to recompute the chain of his tag.

$$\begin{array}{cccccccc}
S_1^0 & \longrightarrow & r_1^0 & r_1^1 & r_1^2 & \dots & r_1^{L-1} & r_1^L \\
S_2^0 & \longrightarrow & r_2^0 & r_2^1 & r_2^2 & \dots & r_2^{L-1} & r_2^L \\
\dots & \longrightarrow & \dots & \dots & \dots & \dots & \dots & \dots \\
S_i^0 & \longrightarrow & \dots & \dots & \dots & \boxed{r_i^j = \mathcal{G}(\mathcal{H}^j(S_i^0))} & \dots & r_i^L \\
\dots & \longrightarrow & \dots & \dots & \dots & \dots & \dots & \dots \\
S_n^0 & \longrightarrow & r_n^0 & r_n^1 & r_n^2 & \dots & r_n^{L-1} & r_n^L
\end{array}$$

Fig. 2. OSK table: chains of hashes in the OSK protocol.

Beside this desynchronization issue – and although the protocol is very efficient when all the tags are synchronized [20] – the worst-case complexity of the search procedure makes the protocol unsuitable for most practical applications.

2.2 Real-Life Applications

We now discuss the possibility of implementing OSK in real-life applications. Throughout this article, we chose a system of $n = 2^{20}$ tags with a lifetime of $L = 2^7$, which are reasonable parameters and in accordance with [3].

Online Search. A naïve approach for the server is to only keep the initial secrets and recompute the $n \times L$ possibilities each time it receives a given σ. With a server capable of 2^{20} cryptographic hash operations per second, this takes 2^6 s ≈ 1 min on average for these parameters. This is far beyond our limit of 200 ms for a reasonable identification time.

Full Storage. The other extreme solution consists in storing all the chains in a table and letting the server perform a simple look-up whenever it receives σ. This solution has the advantage of requiring no cryptographic operation during the authentication, which makes the authentication very fast. Unfortunately, this approach has two major drawbacks.

First of all, a large memory is needed to store the table: given our parameters ($n = 2^{20}$ tags with a lifetime of $L = 2^7$, and a hash size of 128 bits), the full storage approach requires 2^{34} bits = 2 GB.[3] In a system where readers are permanently connected to the back-end server, requiring such a memory (RAM) for the server is not a major problem. However, in systems consisting of mobile readers sporadically connected to the database, the authentication material should be replicated in each of these low-resource devices. In such a scenario, this amount of memory is very large for small devices such as PDA's or handheld RFID readers, which typically have a memory of 128 MB. It might, however, be reasonable for more elaborate devices such as NFC-enabled smartphones, which have several gigabytes of flash memory.

A second issue is that, every now and then, the table needs to be either computed in a central server and uploaded on the smartphones, or computed by the smartphones themselves after reception of the first column of the table. This might take a significant time for both cases, and might be an issue in certain situations.

In the context of [6] for instance, where a central server is used, the full storage technique makes sense, and is more simple and efficient.

Time-Memory Trade-off. An intermediate solution is the time-memory trade-off (TMTO). The idea is to use memory to reduce the authentication time, making both memory and time suitable to our application. Note that the goal of the TMTO is here to reach an authentication time that is below the acceptable threshold of 200 ms. Once this requirement is fulfilled, still decreasing the time

[3] If one wants to index the hashes with (i, j) couples, the memory increases by 25 % (32 bits appended to each of the 128-bit hashes).

Table 1. Comparison of exhaustive search and table look-up methods (Average case).

	Exhaustive search	Exhaustive storage
Precomputation	0	N
Online computation	$N/2$	0
Memory (storage)	0	N

does not make sense because (i) this implies a memory cost (ii) the authentication time would become a negligible factor in the whole communication time.

3 Background on Time-Memory Trade-offs

In this section, we briefly recall the required background on the *Time-Memory Trade-off* (TMTO) method. We describe the TMTO technique but make no attempt at providing a complete survey of it. For an advanced introduction about this topic we recommend to read [4].

3.1 Introduction

A common search problem in cryptanalysis is finding the preimage of a given output of a one-way function. The first naïve method is applying the function to all possible inputs until finding the expected value. Such an exhaustive search requires N operations in the worst case to find a preimage, where N is the total size of the problem. This becomes impractical when N is large.

The other extreme is to first construct a look-up table including all the preimage values. Afterwards, finding a preimage is done via a table look-up operation which requires a negligible amount of time. The precomputation process requires an effort equal to an exhaustive search, but is to be performed only once. Although this method is quite fast during the online search phase, it may require extreme amounts of memory for large problems.

The comparison of exhaustive search and exhaustive storage methods is depicted in Table 1.

3.2 Description

The basic idea of the time-memory trade-off (TMTO) method is to find a compromise that has a lower online computation complexity than the exhaustive key search, and a lower memory complexity than the exhaustive storage. Hellman introduced one such trade-off in 1980 [13]. Given a search space of size N, and given M words of memory used for the trade-off, the average number of cryptographic operations T obeys the law $N^2 \propto T \times M^2$ [13]. The principle is the following. During an initial phase, a point is chosen arbitrarily in the search space, hashed (or ciphered, depending on the target function), and then reduced to another point in the search space. This reduction is the output of a reduction function, which is typically a modulo. This process is iterated a given number of times, forming a chain of hashes. This whole operation is itself repeated many times and only the starting points and endpoints of the chains are kept and stored in a table. Once this table is computed, it is used in the online phase to accelerate the search. The method is probabilistic given that it is very unlikely to fully cover the search space, but several tables can be used to obtain a success probability very close to 1.

A major improvement over Hellman's original TMTO method [13] was given by Oechslin in [8]. The precomputed table called *rainbow table* for this method is structurally different than Hellman's TMTO in that it uses a different reduction function in each column. By doing so, although it might seem to slow the search process, chain fusions (events in the table construction and the search process that degrade the efficiency) in the table are much less frequent and can be detected very easily during their construction. Tables without fusion are said *perfect* [9] and will be used in this paper.

We now give the most relevant results in the analysis of rainbow tables.

Theorem 1 *The probability of success of a set of ℓ rainbow tables of m rows of t columns each, for a problem of size N is:*

$$P = 1 - \left(1 - \frac{m_t}{N}\right)^{\ell t}.$$

Proof See [4].

Theorem 2 *The maximum number of chains in a rainbow table of t columns, for a problem of size N is:*

$$m_t^{max} = \frac{2N}{t+1}.$$

Proof See [4].

Theorem 3 *The optimal parameters for a rainbow table, for a problem of size N, given a memory of M and a desired probability of success P^* are:*

$$\begin{aligned} \ell &= \left\lceil \frac{-\log(1-P^*)}{2} \right\rceil, \\ m_t &= \frac{M}{\ell}, \\ t &= \frac{\log(1-P^*)}{\ell \log\left(1 - \frac{m_t}{N}\right)} \approx -\frac{N}{M}\log(1-P^*). \end{aligned}$$

Proof See [4].

In the following, optimal parameters are implicitly used.

4 OSK/AO

4.1 Description

Avoine and Oechslin propose in [5] to apply the time-memory trade-offs to the search procedure of OSK, leading so to a variant known as OSK/AO. The complexity of the search procedure varies from $O(1)$ to $O(N)$, depending on the amount of memory we are willing to devote to the time-memory trade-off. For example, they mention that a complexity of $O(N^{2/3})$ can be reached with a memory of size $O(N^{2/3})$.

Avoine, Dysli, and Oechslin also suggest in [3] a variant of the OSK protocol that ensures strong authentication, as OSK is originally designed to ensure identification only, without consequently considering replay attacks. To do so, [3] suggests using nonces as follows: the reader sends a nonce r in the authentication request message and the tag answers $\mathcal{G}(S_i^j \oplus r)$ along with $\mathcal{G}(S_i^j)$. The latter value is used by the reader to identify the tag, and the former to authenticate it.

Another advantage of OSK/AO is that the search done in the identification is intrinsically randomized, which makes timing attacks irrelevant [2].

Now we briefly describe the specific time-memory trade-off technique introduced in [3,5].

In this technique there are two main functions namely a *response generating function* $\mathcal{F}$ and a *reduction function* $\mathcal{R}$. $\mathcal{F}$ takes two indices as an input (i.e., tag index and life time index) and outputs a tag response such that

$$\mathcal{F} : (i, j) \mapsto \mathcal{G}(\mathcal{H}^j(S_i^0)) = r_i^j$$

The reduction function $\mathcal{R}$ is such that

$$\mathcal{R} : r_i^j \mapsto (i', j')$$

where $1 \leq i, i' \leq n$, and $0 \leq j, j' \leq L$.

The main specificity is that $\mathcal{F}$ requires $j + 1$ cryptographic operations to be computed, which would drastically lower the efficiency of the search if it were used directly. What is suggested instead is to use a second kind of-time-memory trade-off, called the rapid-hash table, to compute $\mathcal{F}$ efficiently. This trade-off table is rather straightforward: the secrets S_i^j of the tags are computed from life-time values 0 to L, but only $\frac{L}{\kappa}$ columns are stored. This is illustrated in Fig. 3. As explained in Sect. 4.3, this means that an average of $\frac{\kappa+1}{2}$ cryptographic operations are required per evaluation of $\mathcal{F}$.

4.2 Analysis

As explained in Sect. 4.1, there are two things that need to be stored in memory: the rainbow tables and the rapid-hash table. We discuss below the proportion of memory that should be dedicated to each.

Let ρ denote the proportion of memory dedicated to the rainbow tables. The trade-off efficiency follows the rule $T = N^2\gamma/M_{RT}^2$ (see [4,13]), with γ being a

$$\underbrace{\begin{array}{cccccc}
S_1^0 & S_0^\kappa & S_0^{2\kappa} & \dots & S_0^{(\lfloor \frac{L}{\kappa} \rfloor -2)\kappa} & S_0^{(\lfloor \frac{L}{\kappa} \rfloor -1)\kappa} \\
S_2^0 & S_1^\kappa & S_1^{2\kappa} & \dots & S_1^{(\lfloor \frac{L}{\kappa} \rfloor -2)\kappa} & S_1^{(\lfloor \frac{L}{\kappa} \rfloor -1)\kappa} \\
\vdots & & & & & \vdots \\
S_i^0 & S_i^\kappa & S_i^{2\kappa} & \dots & S_i^{(\lfloor \frac{L}{\kappa} \rfloor -2)\kappa} & S_i^{(\lfloor \frac{L}{\kappa} \rfloor -1)\kappa} \\
\vdots & & & & & \vdots \\
S_n^0 & S_{n-1}^\kappa & S_{n-1}^{2\kappa} & \dots & S_{n-1}^{(\lfloor \frac{L}{\kappa} \rfloor -2)\kappa} & S_{n-1}^{(\lfloor \frac{L}{\kappa} \rfloor -1)\kappa}
\end{array}}_{\frac{L}{\kappa} \text{ columns}}$$

Fig. 3. The rapid-hash table.

small factor depending on the probability of success of the trade-off, and M_{RT} the memory dedicated to the rainbow tables (that is ρM). As for the rapid-hash table, we have:

$$\kappa = \left\lceil \frac{N|hash|}{M_{RH}} \right\rceil,$$

with $|hash|$ the size of a hash, and M_{RH} the memory for the rapid-hash table (that is $(1-\rho)M$). Each operation in the rainbow tables requires an average of $\frac{\kappa+1}{2}$ cryptographic operations in the rapid-hash table. Therefore:

$$T = \frac{N^2\gamma}{M_{RT}^2}\frac{\kappa+1}{2} \approx \frac{N^2\gamma}{\rho^2 M^2}\frac{N|hash|}{2(1-\rho)M}.$$

The optimal value of ρ can be found easily by deriving:

$$\frac{\partial T}{\partial \rho} = 0 \quad \Leftrightarrow \quad \frac{\partial}{\partial \rho}\left[\frac{1}{\rho^2(1-\rho)}\right] = \frac{3\rho-2}{(\rho-1)^2\rho^3} = 0,$$

which yields $\rho_{opt} = \frac{2}{3}$. In the following, we will thus take the memory for the rainbow tables to be two thirds of the total memory.[4]

4.3 Algorithms

We now describe the algorithms used in OSK/AO, namely (i) the algorithm to compute the rapid-hash table (Algorithm 1), (ii) the algorithm to build the TMTO tables (Algorithm 2), and (iii) the algorithm to identify the tag (Algorithm 3). The material in this section mostly comes from [6]. The notations used in the algorithms are given in Table 2.

First, the system randomly generates the initial secrets for all the tags such that $S_i^0 \in_R \{0,1\}^\lambda$ where $1 \le i \le n$, and λ is the length of the secrets. The

[4] Note that this result is compliant with the analysis done in [5]. The development done in this section is somewhat simpler and matches the notations used in the rest of this paper.

Table 2. Notations used throughout the paper.

n	Number of tags in the system
L	Life time of a tag in the system (in terms of authentication executions)
ℓ	Number of TMTO tables
t	Length of the chains of a rainbow table
S_i^j	Secret of the ith tag used for the $j+1$th authentication where $1 \leq i \leq n$, and $0 \leq j \leq L$
$\mathcal{H}$, $\mathcal{G}$	Collision resistant one-way functions
$rapid\mathcal{H}(i,j)$	Function which computes the jth secret of the ith tag such that $S_i^j = \mathcal{H}^j(S_i^0)$. This function uses a precomputed rapid hash (RH) table to compute hashes faster. The construction of this function is demonstrated in Algorithm 1
κ	Length of the interval between hash indices. This parameter is needed for computing rapid hashes
$state[i][k]$	A pre-computed two-dimensional array which stores the $k \times \kappa$th hash value of the ith tag's initial secret $(\mathcal{H}^{k\times\kappa}(S_i^0))$. For instance, let $\kappa = 6$, $i = 1$ and $k = 6$, then $state[1][6]$ stores $S_1^{36} = \mathcal{H}^{36}(S_1^0)$. This array is used during the evaluation of $rapid\mathcal{H}(i,j)$
$\mathcal{F}(i,j)$	The response generating function inputs two parameters, the tag index and the life time of the tag. This function uses the rapid-hash function. It outputs a tag response such that $\mathcal{F}(i,j) = \mathcal{G}(rapid\mathcal{H}(i,j))$
$Table_v$	The vth TMTO table which stores the starting and endpoints (indices) of the TMTO table, where $1 \leq v \leq \ell$
$\mathcal{R}_w^v(val)$	For wth column of the vth table, a simple reduction function which maps input val into a output with smaller size, where $1 \leq w \leq t$

system defines a κ parameter then computes the interval secret values of all the tags. After that all the secrets are stored into a two dimensional array such that $state[i][k] := \mathcal{H}^{k\times\kappa}(S_i^0)$ where $k = 0, 1, 2, \ldots$ and $0 \leq k \times \kappa \leq L$.

Now, for a given secret of tag i, the jth rapid-hash computation of the secret is presented in Algorithm 1. The algorithm requires only at most κ hashes by

Algorithm 1 Compute $y = rapid\mathcal{H}(i,j)$

```
Require: $1 \leq i \leq n, 0 \leq j \leq L$
Ensure: $y = S_i^j$
  $y \leftarrow state[i][\lfloor \frac{j}{\kappa} \rfloor]$
  $a \leftarrow j \bmod \kappa$
  while $a \neq 0$ do
    $y = \mathcal{H}(y)$
    $a \leftarrow a - 1$
  end while
  return $y$
```

the help of the precomputed RH table. Whenever κ decreases, the memory usage increases but the on-line computation decreases.

Algorithm 2 shows the procedure to construct a single rainbow TMTO table. For the construction, only two parameters are needed: the number of starting points used in the precomputation phase (generally named m_1 [4]) and the number of the table to be generated. The starting points of a TMTO table are fed into the $\mathcal{F}$ function sequentially. The output is actually a response of a tag in the system and is fed into the reduction function which outputs arbitrary indices. For a single chain this process is repeated consecutively up to a pre-defined chain size t, then the starting and endpoints are stored in the table. Finally, each generated ending point is compared in the table to detect fusions. When two chains generate a fusion, one of them is discarded. This procedure eventually leads to a perfect table.

Algorithm 2 Construction of $Table_v\ (j,\, m_1,\, v)$

```
Require: $1 \leq j,\ 1 \leq m_1 \leq n \times j\ ,\ v \geq 1$
  $table \leftarrow \{\emptyset\}$
  for $i = 1$ to $\left\lceil \frac{m_1}{j} \right\rceil$ do
    for $k = 0$ to $j$ do
      $nextResp \leftarrow \mathcal{F}(i, k)$
      for $w = 1$ to $t - 1$ do
        $z[\ ] \leftarrow \mathcal{R}_w^v(nextResp)$
        $nextResp = \mathcal{F}(z[0], z[1])$
      end for
      $z[\ ] \leftarrow \mathcal{R}_t^v(nextResp)$
      if $z \notin table$ then
        add the record $\{(i, k); (z[0], z[1])\}$ into $table$
      end if
      if $(i - 1) \times j + k \geq m_1$ then
        break
      end if
    end for
  end for
  clean $table$
  return $table$
```

Finally, Algorithm 3 shows the identification process of a tag by extracting the pre-image of a given response using TMTO tables. This part of the system runs during the authentication of a tag. First, $TagResp$ (the answer of the tag) is fed into the reduction function R_t^v and searched among the ending points of the TMTO table. (i) If a match is found, the corresponding starting point is iterated as explained in Algorithm 2 up to the $(t-1)$th reduction function R_t^v in order to get a candidate response. If the candidate response is equal to $TagResp$ then identification is completed. Otherwise (ii) $TagResp$ fed into the reduction function such that $R_{t-1}^v(TagResp)$, then the resulting indices fed into

$\mathcal{F}$, and then the resulting response fed into $R_t^v(TagResp_{next})$ consecutively. As previously done, the output value search among the endpoints of the TMTO table and the similar process is carried as described above.

Algorithm 3 Identify ($Table_v$, TagResp)

```
Require: TagResp $\in \{0,1\}^\lambda$, $v \geq 1$
Ensure: TagResp $\leftarrow \mathcal{G}(y)$
  for $q = t$ down to 1 do
    $nextResp \leftarrow$TagResp
    for $i = q$ to $t - 1$ do
      $z[\,] \leftarrow \mathcal{R}_i^v(nextResp)$
      $nextResp \leftarrow \mathcal{F}(z[0], z[1])$
    end for
    $z[\,] \leftarrow \mathcal{R}_t^v(nextResp)$
    if $z \in Table_v$ then
      $\{z'; z\} \leftarrow Table_v(z)$
      $nextResp \leftarrow \mathcal{F}(z'[0], z'[1])$
      for $w = 1$ to $q - 1$ do
        $\widetilde{z}[\,] \leftarrow \mathcal{R}_w^v(nextResp)$
        $nextResp \leftarrow \mathcal{F}(\widetilde{z}[0], \widetilde{z}[1])$
      end for
      if $nextResp = TagResp$ then
        return $true$
      end if
    end if
  end for
  return $false$
```

5 Experiments and Comparison

5.1 Environment

The precomputations are performed with a personal computer having Intel 2.8 GHz Core2 Duo processor, 4 GB RAM and Windows 7 - 64-bit operating system. As an NFC enabled mobile phone we use LG OPTIMUS 4X HD having 1.5 GHz processor and 1 GB RAM [17]. The cell phone has an open source Linux-based operating system, Android. This OS has a large community of contributors who develop applications primarily written in a customized version of the Java programming language [22]. The phone supports both ISO/IEC 14443 and ISO/IEC 15693 standards which are the common standards in order to read/write 13.56 MHz contactless smart cards.

For the tags, we work on professional version of ZeitControlers basic card $ZC7.5\,(ZC - Basic)$ which is a programmable processor card as hardware environment for protocol implementation [12]. It has a micro-controller with 32 kB user EEPROM that holds its own operating system (OS) and it has 2.9 kB

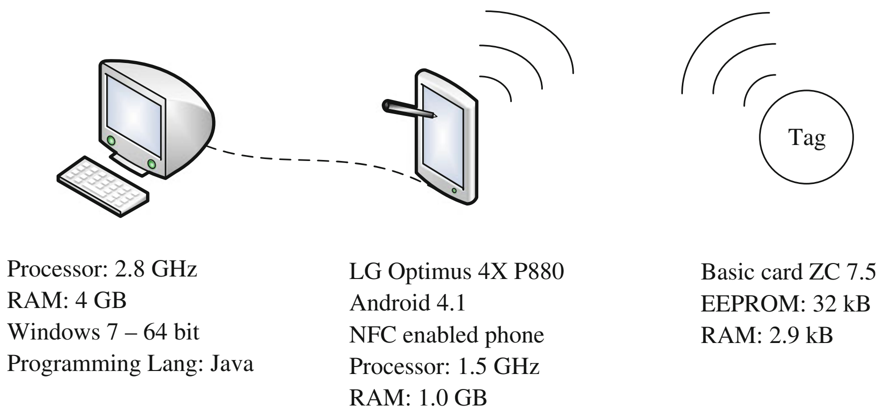


Fig. 4. Overview of the system

RAM for user tag's data. It supports ISO/IEC 14443. The EEPROM contains the user's Basic code, compiled into a virtual machine language known as P-Code (the Java programming language uses the same technology). The RAM contains run-time data and the P-Code stack. The overview of the system is depicted in Fig. 4.

5.2 Parameters and Functions

The parameters for the experiments are $n = 2^{20}, L = 2^7$ and the one-way functions we selected are the following ones[5]:

- $\mathcal{H}(S_i^j) : AES_K(S_i^j) \oplus S_i^j = S_i^{j+1}$,
- $\mathcal{G}(S_i^j) : AES_K(S_i^j + 1) \oplus (S_i^j + 1) = r_i^j$

where K is a 128-bit constant key. This is known as the Matyas-Meyer-Oseas construction [19]. Its goal is to build a one-way function from a block cipher.

We use the AES algorithm in the construction because it is commonly implemented on fewer gates than classical hash functions (see e.g. [11]), and, in particular, is also available in the ZC7.5. This construction requires only one key schedule during the initialization phase of the tags, which makes algorithm faster.

To construct rainbow tables each column of each table uses a different reduction function. The function takes three parameters that are the table index ($v \in [0, 1, \ldots, \ell - 1]$), the column index ($w \in [1, 2, \ldots, t]$) and the response output as a byte array ($val[.]$). This function produces two output values; the first one is for tag index ($i = 0, \ldots, n - 1$), the second one is for lifetime index ($j = 0, \ldots, L - 1$). The i value is computed as $i = (Int32(val[v, v + 3]) + w)$

[5] The parameters are the same than the ones in [3].

mod n where the function $Int32$ converts a given input 4-byte array into an unsigned 32-bit integer. The j value is computed as $i = (Int32(val[v+1, v+4])+w)$ mod L. The construction of our reduction functions are given in Algorithm 4.

Algorithm 4 Compute $\mathcal{R}^v_w(val[.])$

```
**Require:** $v \geq 0$, $w \geq 1$
**Ensure:** $i \in \mathbb{Z}_n$, $j \in \mathbb{Z}_L$
  $i \leftarrow Int32(val[v, v+3]) + w$
  $j \leftarrow Int32(val[v+1, v+4]) + w$
  $i = i \ mod \ n$
  $j = j \ mod \ L$
  **return** $\{i, j\}$
```

5.3 Precomputation of the Tables

In order to use our implementation on low-resource devices (such as hand-held readers, PDAs and NFC compliant cellphones) we build tables that can fit to small RAMs.

For the total memory there are two parts: (i) the rapid hash table that stores some intermediate values of the OSK table and (ii) the TMTO tables.[6] We use the optimal parameters, so we compute the κ and t such that the memory consumption is as described in Sect. 4.2.

Another significant choice for the TMTO construction is the probability of success. It should be high enough to avoid false negatives during the authentication process. In our scenario, using $\ell = 4$ rainbow tables of maximal size, the probability to identify a tag is greater than 0.999 according to Theorem 1. Note that trying to reach a higher success probability does not make sense given that the probability of failure due to noise on the channel is even higher.

Finally, regarding the number of starting points m_1, we use the same trick as in [4] to reduce the precomputation effort. In our case, we obtain about 98 % of the maximal number of ending points by starting with 50 times that number.

In total, the precomputation cost is $\ell \times m_1 \times t$ evaluations of $\mathcal{F}$, which is about $4 \times 50 \times m_t \times t = 400\,nL$ in our case (see Theorem 2). Since these are $\mathcal{F}$ evaluations, this number is also multiplied by $\frac{\kappa+1}{2}$ hash operations. For instance, if $\kappa = 6$ and on a server capable of 2^{20} hash operations per second, the precomputation stage would take about 50 h. Some details about the precomputation of rainbow tables seem to have been overlooked in [3,5], which would explain their optimistic result. However, we can do much better than that if we build a table containing the nL secrets, and use it during the precomputation instead of the $rapidH$ table. This table needs $nL|hash|$ bits, that is 2 GB in our case, and takes about 2 min to build on the server. In this case, there are actually no hash operations during the building of the TMTO table, making this procedure faster. In our case the whole precomputation process takes about an hour.

[6] We used the prefix-suffix decomposition method, as described for instance in [7] in order to reduce to some extent the size of the TMTO tables.

Table 3. Results of experiments on an NFC compliant cellphone

Memory	253 MB	113 MB
Identification time	15.26 ms	117.54 ms
Length of the chains of the TMTO (t)	27	72
Number of chains of the TMTO (m_t)	8968214	3566605
Rapid-hash parameter (κ)	22	43
Authentication rate	99.9 %	99.9 %

5.4 Experiments

We tested the performance in two settings by running Algorithm 3 (i.e., identification process of OSK/AO with randomly chosen tags). Our mobile phone [17] is able to compute about 187,750 hashes per second. For both settings, the experiment is run 1,000,000 times. The experimental results are depicted in Table 3.

We also measure the time when we use our system with a real tag. There are three phases on the tag's side: receiving a query, computing the response (two hash calculations), and sending the response. The total time is 70 ms on average, including 50 ms for the calculation of the two hash values and 20 ms for the communication.

It can be seen that the average identification time is below the 200 ms threshold (if we include the 70 ms for the tag computation and the communication) even for a memory below 128 MB. We thus show that one can achieve very fast authentication even with limited memory.

6 Conclusion

We have implemented the OSK/AO [3] protocol on an NFC-compliant cellphone and a ZC7.5 contactless tag. Our implementation is fully operational and is, to the best of our knowledge, the first implementation of a privacy-friendly authentication protocol based on symmetric-key cryptography. The implementation is suited to large-scale applications, e.g. a million of tags, as this can be the case in mass transportation systems, even on low-resource mobile devices such as hand held readers, PDAs or NFC compliant cellphones. We have run several experiments on the implemented RFID system and we show that the results obtained match the theory and are favorable to a practical deployment.

Acknowledgements. This work is partially funded by the Walloon Region Marshall plan through the SPW DG06 Project TRASILUX.

References

1. Avoine, G., Bingöl, M.A., Carpent, X., Ors Yalcin, S.B.: Privacy-friendly authentication in RFID systems: on sub-linear protocols based on symmetric-key cryptography. IEEE Trans. Mob. Comput. **12**, 2037–2049 (2013)

2. Avoine, G., Coisel, I., Martin, T.: Time measurement threatens privacy-friendly RFID authentication protocols. In: Ors Yalcin, S.B. (ed.) RFIDSec 2010. LNCS, vol. 6370, pp. 138–157. Springer, Heidelberg (2010)
3. Avoine, G., Dysli, E., Oechslin, P.: Reducing time complexity in RFID systems. In: Preneel, B., Tavares, S. (eds.) SAC 2005. LNCS, vol. 3897, pp. 291–306. Springer, Heidelberg (2006)
4. Avoine, G., Junod, P., Oechslin, P.: Characterization and improvement of time-memory trade-off based on perfect tables. ACM Trans. Inf. Syst. Secur. **11**, 17:1–17:22 (2008)
5. Avoine, G., Oechslin, P.: A scalable and provably secure hash based RFID protocol. In: International Workshop on Pervasive Computing and Communication Security - PerSec 2005, Kauai Island, HI, USA, March 2005, pp. 110–114. IEEE Computer Society (2005)
6. Bingöl, M.A.: Security analysis of RFID authentication protocols based on symmetric cryptography and implementation of a forward private scheme. Master's thesis, Istanbul Technical University, Istanbul, Turkey (2012)
7. Biryukov, A., Shamir, A., Wagner, D.: Real time cryptanalysis of A5/1 on a PC. In: Schneier, B. (ed.) FSE 2000. LNCS, vol. 1978, pp. 1–18. Springer, Heidelberg (2001)
8. Oechslin, P.: Making a faster cryptanalytic time-memory trade-off. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 617–630. Springer, Heidelberg (2003)
9. Borst, J., Preneel, B., Vandewalle, J.: On the time-memory tradeoff between exhaustive key search and table precomputation. In: Proceeding of the 19th Symposium in Information Theory in the Benelux, WIC, Veldhoven, The Netherlands, pp. 111–118 (1998)
10. HID Global Corporation. HSPD-12 & FIPS 201 PIV II: How Government Standards Affect Physical Access Control. http://www.hidglobal.com/sites/hidglobal.com/files/hid-how-gov-stanards-affect-physical-access-control-wp-en.pdf (2007)
11. Feldhofer, M., Wolkerstorfer, J., Rijmen, V.: AES implementation on a grain of sand. IEE Proc.: Inf. Secur. **152**(1), 13–20 (2005)
12. Guilfoyle, T.: The zeitcontrol basiccard family. http://www.basiccard.com (2009)
13. Hellman, M.: A cryptanalytic time-memory trade-off. IEEE Trans. Inf. Theory **26**(4), 401–406 (1980)
14. International Organization for Standardization. ISO/IEC 9798: Information technology - Security techniques - Entity authentication - Part 2: Mechanisms using symmetric encipherment algorithms (1999)
15. Juels, A., Weis, S.: Defining strong privacy for RFID. In: International Conference on Pervasive Computing and Communications - PerCom 2007, March 2007, pp. 342–347. IEEE Computer Society, New York (2007)
16. Kardaş, S., Levi, A., Murat, E.: Providing resistance against server information leakage in RFID systems. In: New Technologies, Mobility and Security - NTMS'11, Paris, France, February 2011, pp. 1–7. IEEE Computer Society (2011)
17. LG Optimus 4X HD P880. Technical Specifications. http://www.lg.com/uk/mobile-phones/lg-P880/technical-specifications (2013)
18. Lim, C.H., Kwon, T.: Strong and robust RFID authentication enabling perfect ownership transfer. In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 1–20. Springer, Heidelberg (2006)
19. Matyas, S.M., Meyer, C.H., Oseas, J.: Generating strong one-way functions with cryptographic algorithm. IBM Tech. Discl. Bull. **27**(10A), 5658–5659 (1985)
20. Ohkubo, M., Suzuki, K., Kinoshita, S.: Cryptographic approach to privacy-friendly tags. In: RFID Privacy Workshop. MIT, Cambridge (2003)

21. Phan, R.C.-W., Wu, J., Ouafi, K., Stinson, D.R.: Privacy analysis of forward and backward untraceable rfid authentication schemes. Wirel. Pers. Commun. **61**(1), 69–81 (2011)
22. Shankland, S.: Google's Android parts ways with Java industry group. CNET News (November 12, 2007). Accessed 15 Feb 2012

Is NFC a Better Option Instead of EPC Gen-2 in Safe Medication of Inpatients

Mehmet Hilal Özcanhan, Gökhan Dalkılıç(✉), and Semih Utku

Department of Computer Engineering, Dokuz Eylul University, Izmir, Turkey
{hozcanhan, dalkilic, semih}@cs.deu.edu.tr

Abstract. Wrong medication is an important problem of the citizens of many countries. Using contemporary technologies like UHF Gen-2 RFID tags helps decreasing the medication errors, experienced by many. As UHF Gen-2 tags have limited capacity, cryptographic algorithms cannot be accommodated. The only available functions PRNG and CRC cannot be used instead of cryptographic algorithms. To overcome the known weaknesses, various grouping protocols have been proposed. But, each protocol has some deficiencies. Two of those protocols are covered in this study. Some of their common deficiencies are studied and solutions are suggested.

Keywords: Patient safety · Medication error · RFID · UHF tag · EPC Gen 2 · NFC · ISO 18000-6 · Authentication · Group proofing protocol

1 Introduction

It is reported that 78 % of the participants in the Eurobarometer survey, on the perception of medical errors, have voted wrong medication as an important problem, in their country [1]. The poll indicates that 23 % of the participants have been directly or indirectly affected, by a medical error. 18 % reported that they experienced a serious medical error, in a hospital. This contradicts the major patient safety goal of avoiding harm caused, during medical care [2]. The need for better patient safety is stated in many works [3,4].

The medication error definition is given as errors in drug ordering, transcribing, dispensing, administering, or monitoring [2]. This work is concerned with correct drug administering of an inpatient, at the correct time; i.e. drug administration free of humanerrors due to patient-drug pack mismatch. Many technologies are used in hospital automation systems, from high-end servers, to personal digital assistants (PDA), tablets, automatic medicine dispensers (AMD) and recently radio frequency identification (RFID) tags. Doctors and nurses are the users of these technologies and they have tablets, which they know how to use. Some of these tablets even have an integrated tag reader. On the other hand, the patients are the subjects who need to be tracked, correctly. RFID tags are one of the best tools available for identification and tracking of subjects. For

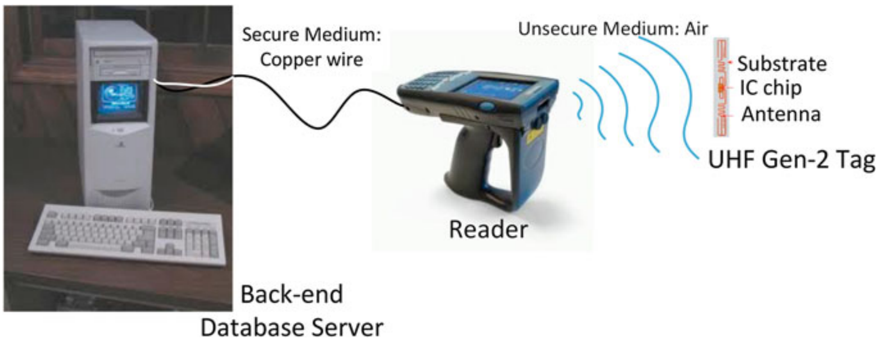


Fig. 1. A typical UHF RFID tag reading scenario

example, one of the biggest chain stores of U.S.A., the Walmart, started using RFID tags on its goods, in 2005 [5]. Walmart has gradually replaced traditional paper barcodes with RFID tags. Recently, passive UHF RFID tags have been proposed in inpatient medication. Passive tags are named as such because; they have no battery but are energized by the reader that approaches to read the ID inside the tag. These tags are used as bracelets for inpatients and as tags on medicine packs. Passive UHF tags are preferred because of their low cost and long reading distance, but they have limited resources and lack security primitives. A specific type of passive UHF tag can be read from a few meters. As many as hundreds of tags, can be read per second. According to ISO 18000-6 and EPC Global Class 1 Generation 2 (Gen-2) standards written for UHF tags, they contain only a 16 bit pseudo random number generator (PRNG), a CRC and an XOR function to obscure their messages [6,7]. Therefore, the capture of the Electronic Product Code (EPC), i.e. the ID of the tag, is not difficult.

A typical RFID set up used in patient identification consists of a back-end database server (server), a reader and a tag (Fig. 1). The server has all the information about a subject: personal information, the unique identification number (ID) of the inpatient's wristband tag, the ID of the tag on the inpatient's medicine pack and the pre-shared secrets used for authentication (also stored in the tags).

In the rest of this paper, Sect. 2 summarizes previous work. Sections 3 and 4 demonstrates weaknesses of two latest proposals. Section 5 questions the use of Gen-2 tags and, proposes another type of tag that is better suited for healthcare. Some critical capabilities and characteristics of the two tag types are also compared, in Sect. 5. Section 6 concludes and has the future work.

2 Related Work

Juels et al.'s work [8] is one of the first in identification of a group of objects, using RFID tags. In this work, a grouping proof is defined as the simultaneous reading of two tags at a given timestamp. Other grouping proofs have also been proposed for tags [9]. The weaknesses and recommended security enhancements for grouping proofs in general can be found in [10].

Two of the first proposals to use RFID tags in patient medication were made by Wu et al. and Sun et al. [11,12]. These pioneering work in inpatient medicine administration, lacked detailed description and advocated the use of personal computers as mobile devices and paper barcodes. Barcodes have limited capabilities and there are disadvantages of their use in patient safety [13].

A proposal where both the inpatient and the medicine are identified by low-cost RFID tags conforming to Gen-2 standard was made by Huang and Ku [14]. The inpatient is assumed to have a wristband with an embedded RFID tag. The inpatient's medicine container is also marked with a Gen-2 tag. Unfortunately, the security flaws in the grouping proof are demonstrated by Chien et al. [15]; whom suggested an alternative protocol, which is shown to be also vulnerable [16]. The grouping proof protocol schemes, suggest evidence to be generated after the administration of medicine. The evidence is verified later, in the HIS server. False evidence generation, interference with evidence generation procedure, exposure of critical information during evidence creation are some of the problems encountered. The issues arise, because the unsecured messages through the air are eavesdropped by adversaries.

Apart from the above, we demonstrate further weaknesses in two recent works, in Sects. 3 and 4. The two works are specifically chosen because, their authors try to rectify previous vulnerable schemes, but fail because, they do not consider the algebraic attacks outlined in previous works due to non-availability of security primitives, in Gen-2 tags. The works fail because, they do not consider the enhancements neither in [10], nor the algebraic attacks outlined in [17].

3 Case Study I

Our assumptions for both case studies are as follows. While the reader and the server communicate over a secure channel, the tag and the reader's channel is insecure. The tag has limited resources but the reader has unlimited resources. Therefore, the reader is assumed to support cryptographic algorithms but the tag cannot. The reader is not trusted and a counterfeit reader can be used in the system. Another assumption is that our attacker can listen to the messages between the tag and the reader over the air. The final assumption is the attacker has only passive attack abilities.

The work by Yen et al. analyses some weaknesses of a previous work [16]. The analyzed proposal is the Inpatient Safety RFID System (IS-RFID) of Peris et al. [18]. Skipping the details, the Safe Drug Administration Procedure (SDAP) and the Evi-dence Generation Procedure (EGP) are analyzed. The inexistence of the pre-shared secrets in the SDAP is criticized, but no attack is demonstrated. The EGP is also criti-cized for not being signed by the inpatient, which allows the hospital to re-generate false evidence without inpatient's awareness. Yen's proposed rectified scheme is shown in Fig. 2. We demonstrate a disclosure attack on Yen's rectified scheme, which also succeeds in IS-RFID. The attack will show

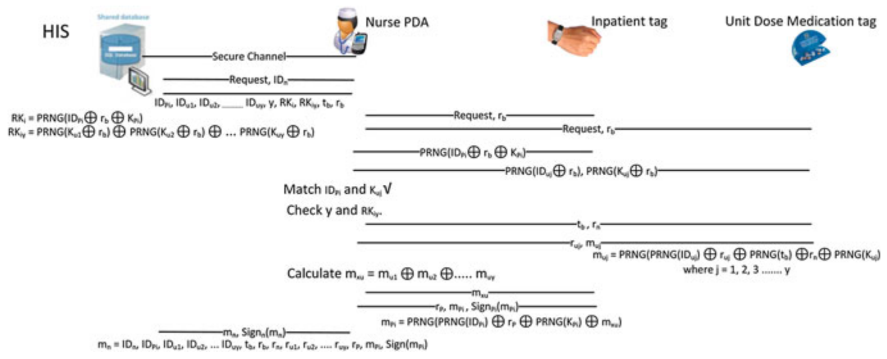

HIS
Shared database
Nurse PDA
Inpatient tag
Unit Dose Medication tag
Secure Channel
Request, ID_n
$ID_{Pi}, ID_{u1}, ID_{u2}, \ldots ID_{uy}, y, RK_i, RK_{iy}, t_b, r_b$
$RK_i = PRNG(ID_{Pi} \oplus r_b \oplus K_{Pi})$
$RK_{iy} = PRNG(K_{u1} \oplus r_b) \oplus PRNG(K_{u2} \oplus r_b) \oplus \ldots PRNG(K_{uy} \oplus r_b)$
Request, r_b
Request, r_b
$PRNG(ID_{Pi} \oplus r_b \oplus K_{Pi})$
$PRNG(ID_{uj} \oplus r_b), PRNG(K_{uj} \oplus r_b)$
Match ID_{Pi} and K_{uj} √
Check y and RK_{iy}.
t_b, r_n
r_{uj}, m_{uj}
$m_{uj} = PRNG(PRNG(ID_{uj}) \oplus r_{uj} \oplus PRNG(t_b) \oplus r_n \oplus PRNG(K_{uj})$
where j = 1, 2, 3 .... y
Calculate $m_{xu} = m_{u1} \oplus m_{u2} \oplus \ldots m_{uy}$
m_{xu}
$r_P, m_{Pi}, Sign_{Pi}(m_{Pi})$
$m_{Pi} = PRNG(PRNG(ID_{Pi}) \oplus r_P \oplus PRNG(K_{Pi}) \oplus m_{xu})$
$m_n, Sign_n(m_n)$
$m_n = ID_n, ID_{Pi}, ID_{u1}, ID_{u2}, \ldots ID_{uy}, t_b, r_b, r_n, r_{u1}, r_{u2}, \ldots r_{uy}, r_P, m_{Pi}, Sign(m_{Pi})$

Fig. 2. Yen's proposed offline scheme [16]

that all protocols based on obscuring the ID of a tag by the use of the PRNG of a Gen-2 tag are vulnerable.

Yen proposes one offline and one online solution. Two schemes are the same, except in the online version, the inpatient's tag is authenticated online by the server, via the nurse PDA. The notation used in Fig. 2 is explained below:

ID_n, ID_{Pi}	ID of a nurse and the tag ID on ith inpatient wristband.
ID_{ui}	Tag ID on a unit dose medicine pack of ith inpatient.
ID_{uj}	Tag ID multiple unit dose medicine packs, j=1,2 . . . y.
K_{Pi}, K_{ui}	Tag key of ith inpatient wristband and ith inpatient's unit dose pack.
K_{uj}	Tag key of multiple unit dose medicine packs, j=1, . . . y.
t_b	Timestamp generated by server.
r_b, r_n, r_p, r_{uj}	Random number generated by server, nurse PDA, inpatient's tag, and jth unit dose, respectively.
PRNG()	16-bit pseudo-random number generation function.
y	Number of unit doses for ith inpatient.
Rk_i	Key validation value for ith inpatient.
Rk_{iy}	Key validation value for ith inpatient's unit doses.
e_i	Evidence generated by a nurse for ith inpatient.
m_{uj}	Partial evidence generated by unit-dose tag j, j=1,2 . . . y.
m_{Pi}	Partial evidence generated by ith inpatient's tag.
m_n	Medication evidence generated by a nurse.
$Sign_n(m_n)$	Signature function of nurse, that signs evidence m_n.
$Sign_{Pi}(m_{Pi})$	Signature function of ith inpatient, that signs evidence m_{Pi}.

Before starting the round, the nurse makes a request with her ID and downloads all inpatient records from the HIS. The daT_a also include the timestamp t_b to supervise the time of drug administration. Validation values RK_i and RK_{iy} are formed by using the pre-shared key of the inpatient's tag and the corresponding unit-dose key, respectively. The nurse starts the round and sends the same request both to the inpatient and the unit-dose tags, with a reader equipped

PDA. Using the HIS nonce r_b, the inpatient tag replies with PRNG($ID_{Pi}\oplus r_b\oplus K_{Pi}$) and every unit-dose tag replies with {PRNG($ID_{uj}\oplus r_b$), PRNG($K_{uj}\oplus r_b$)}, where j=1,2, ...,y. Upon receiving the replies, the PDA matches the inpatient tag's reply with RK_i to identify and authenticate the inpatient. Next, the PDA uses the RK_iy to identify and authenticate the unit-dose packs. If all matching is good, the PDA generates its own nonce r_n and sends it with the timestamp of the HIS to every unit-dose pack. The unit-dose packs generate their own nonce and use it together with nurse PDA's nonce to prepare a partial medication evidence m_{uj} = PRNG[PRNG (ID_{uj}) $\oplus r_{uj}\oplus$ PRNG(t_b) $\oplus r_n\oplus$PRNG (K_{uj})], where j=1, 2, ..., y. Each unit-dose sends back its reply to the nurse PDA. The PDA stores every nonce r_{uj} sent and calculates m_{xu}, by XORing every m_{uj}. The value m_{xu} is sent to inpatient's tag. The inpatient's tag prepares its partial evidence m_{Pi} = PRNG(PRNG(ID_{Pi})$\oplus$ $r_P\oplus$PRNG(K_{Pi})$\oplus m_{xu}$), after generating a nonce reply r_P. Finally the inpatient tag signs its evidence $Sign_{Pi}(m_{Pi})$ and sends the tuple {r_P, m_{Pi}, $Sign_{Pi}(m_{Pi})$} to the nurse PDA. Upon receiving the final partial evidence, the nurse PDA prepares a final medication evidence m_n. The evidence is signed and saved in the PDA, as m_n, $Sign_n(m_n)$. At the end of the round, the nurse returns to the nurse station and uploads all of the drug administration evidence to the HIS. It is the duty of HIS to check and find if there have been any medication errors.

Neither the inpatient's tag nor the nurse PDA digital signature functions are explained. The assumption of inpatient's tag having the computational ability of generating digital signatures is way out of the ISO 18000-6 and Gen-2 standards [6,7]. But, even this assumption cannot save the scheme.

3.1 Disclosure Attack Scenario on Yen's Protocol

The 16 bit PRNG function of the Gen-2 tags is public and available [19]. According to Yen, any PRNG(x) is calculated for a given input x; e.g. using ($ID_{Pi}\oplus r_b\oplus K_{Pi}$) as input, a deterministic output PRNG($ID_{Pi}\oplus r_b\oplus K_{Pi}$) is obtained and matched with R_i. Therefore, a table of 2^{16} (65,536) possible inputs against calculated outputs can be prepared beforehand, as in Table 1. Looking at the table, the corresponding output of an input or the corresponding input of an output can be found, easily. PRNG may produce the same output for the distinct values, but this shows the weakness of the PRNG which is not a desirable property. In that case much more trial and errors are needed.

The inpatient desired to be administered wrong medication is the "*target*". Another inpatient whose identity is going to be illegally given to the target is

Table 1. A typical pre-calculated table

Input	Output = PRNG(Input)
0000 0000 0000 0000	0000 0010 0000 0000
0000 0000 0000 0001	0010 0110 0000 0010
....	....
1111 1111 1111 1111	0100 0111 1100 0110

called the "*conveyor*". The goal is to cause repeated switch of medicine administrations of the target and conveyer, without getting detected. After exposing the $(\mathrm{ID}_{Pi} \oplus \mathrm{K}_{Pi})$ of the conveyer and the target; the identities are switched and detection is avoided.

An adversary/attacker acts as a visitor and goes near the conveyer with a rogue reader. Rogue or untrusted readers are assumed to be always present in open environments [16,18]. The attacker sends a request request, r_a, where r_a is the attacker's constant nonce. The tag answers with $\mathrm{PRNG}(\mathrm{ID}_{Pi} \oplus \mathrm{r}_a \oplus \mathrm{K}_{Pi})$. The output column of the table is searched and the corresponding input; e.g. input1, is found: $\mathit{input1} = (\mathrm{ID}_{Pi} \oplus \mathrm{r}_a \oplus \mathrm{K}_{Pi})$. Then, $(\mathrm{ID}_{Pi} \oplus \mathrm{K}_{Pi}) = \mathit{input}1 \oplus \mathrm{r}_a$. $(\mathrm{ID}_{Pi} \oplus \mathrm{K}_{Pi})$ is constant for any given inpatient; therefore any inpatient is uniquely identified. Using the replies of the unit-dose packs with Table 1 and XORing each $\{\mathrm{PRNG}(\mathrm{ID}_{uj} \oplus \mathrm{r}_a), \mathrm{PRNG}(\mathrm{K}_{uj} \oplus \mathrm{r}_a)\}$ with r_a, all values of $\mathrm{ID}uj$ and $\mathrm{K}uj$ are exposed for j = 1, 2, ..., y. The same attack is repeated at the target. At the end, both the target and conveyor's $(\mathrm{ID}_{Pi} \oplus \mathrm{K}_{Pi})$, $\mathrm{ID}uj$ and $\mathrm{K}uj$ are captured.

Next, the evidence generation procedure of the target and conveyor are eavesdropped for just one round. The messages $\{\mathrm{r}_{uj}, \mathrm{m}_{uj}\}$ of the unit-dose packs in Fig. 2 are recorded, by the attacker. The value of m_{xu}, sent to the conveyor is also recorded. The final reply $\{\mathrm{r}_P, \mathrm{m}_{Pi}, \mathrm{Sign}_{Pi}(\mathrm{m}_{Pi})\}$ of the conveyor is analyzed next. The values not known in $\mathrm{m}_{Pi} = \mathrm{PRNG}(\mathrm{PRNG}(\mathrm{ID}_{Pi}) \oplus \mathrm{r}_P \oplus \mathrm{PRNG}(\mathrm{K}_{Pi}) \oplus \mathrm{m}_{xu})$ are PRNG (ID_{Pi}) and $\mathrm{PRNG}(\mathrm{K}_{Pi})$. But the value $[\mathrm{PRNG}(\mathrm{ID}_{Pi}) \oplus \mathrm{PRNG}(\mathrm{K}_{Pi})]$ is constant and can be exposed. Looking at the output column of Table 1, a match for the value of m_{Pi} is found, e.g. output1. Using output1, $[\mathrm{PRNG}(\mathrm{ID}_{Pi}) \oplus \mathrm{PRNG}(\mathrm{K}_{Pi})] = \mathrm{output1} \oplus \mathrm{r}_P \oplus \mathrm{m}_{xu}$ is obtained. The only unknown left is $\mathrm{Sign}_{Pi}(\mathrm{m}_{Pi})$. The available functions in a Gen-2 tag are PRNG, CRC and XOR operation. Therefore, the assumed out-of-standard, digital signature is most likely to be a deterministic function that has its own 65,536 (216) entry table. Whatever it is, it has to be public and readily available to all tags. Either we have the function and we can construct $\mathrm{Sign}_{Pi}(\mathrm{m}_{Pi})$ out of m_{Pi} or the attacker records the m_{Pi}, $\mathrm{Sign}_{Pi}(\mathrm{m}_{Pi})$ pairs, as a table called Table X. Therefore, the attacker has the m_{Pi}, $\mathrm{Sign}_{Pi}(\mathrm{m}_{Pi})$ pair. The same is repeated near the target.

The attacker takes the exposed values $(\mathrm{ID}_{Pi} \oplus \mathrm{K}_{Pi})$, $[\mathrm{PRNG}(\mathrm{ID}_{Pi}) \oplus \mathrm{PRNG}(\mathrm{K}_{Pi})]$, $\mathrm{Sign}_{Pi}()$ function or Table X, for the target and conveyor and writes them into two different tag emulators, at a private location. Such a hardware device emulating an RFID tag is the Chameleon [20]. We do not intend to implement any, but there are works on RFID tag emulators [21]. The difference from the real tag is that the emulator uses the XORed $(\mathrm{ID}_{Pi} \oplus \mathrm{K}_{Pi})$, $[\mathrm{PRNG}(\mathrm{ID}_{Pi}) \oplus \mathrm{PRNG}(\mathrm{K}_{Pi})]$ values instead of individual values to form its replies.

In the final step of the attack, the tag emulator of the target is placed next to the conveyor and the emulator of the conveyor is placed next to the target. Hence, the switch of the identities is completed. The nurse cannot notice the presence of the switch, because she does not come close to the UHF tags. When the nurse follows normal procedure, the rogue tags generate correct RK_i, $\mathrm{RK}_i y$ and partial evidences. The nurse administers wrong medicine to both patients, signs the evidence and sends them to the HIS. The HIS cannot detect the switch and wrong medication is repeated until the target and impatient show diverse symptoms.

Yen's protocol has another weakness, as well. Blocking $\{\mathrm{t}_b, \mathrm{r}_n\}$ or m_{xu}, and sending bogus instead stop the medication procedure. The medication in a clinic can be disrupted with a strong, bogus transmission.

4 Case Study II

A second work criticizing a previous proposal which uses Gen-2 RFID tags is by Wu et al. [22]. Wu criticizes Yu et al.'s proposal for being based on a fully analyzed protocol [23]. We leave the study of Yu's proposal outside the scope of this work, because we would like to concentrate on Wu's rectified protocol. Wu's proposal is summarized in Fig. 3. The proposal also uses the 16-bit PRNG function of Gen-2 tags to form authenticators and prove the simultaneous existence of two tags, in the same electromagnetic field.

At the beginning the server pre-shares secrets X_a and X_b with tags T_a and T_b, respectively. Typically, the reader challenges both the inpatient and unit-dose tag with the same timestamp, t. Both tags reply with their index-pseudonym (IDS), a nonce and a tag authenticator; $\{\mathrm{IDS}_a, \mathrm{r}_a, \mathrm{v}_a\}$ and $\{\mathrm{IDS}_b, \mathrm{r}_b, \mathrm{v}_b\}$ respectively. The index-pseudonym is a pseudo ID of the tag that is an updated version of constant ID, every round. The reader is online with the back-end server and sends the tag replies together with the timestamp to the server. If verification is good, the server sends two keys K_a, K_b to the reader. Without waiting for a reply, the server immediately updates IDS_a and IDS_b. Using the key Ka, the reader calculates its authenticator α_a and sends $\{\alpha_a, \mathrm{IDS}_b, \mathrm{t}\}$ to tag T_a. T_a calculates its own $\alpha_a^{'}$ and matches it with the received α_a. If they are a match, T_a prepares β_a and partial evidence ma and sends them to the reader. Then, T_a updates. The reader verifies β_a and then prepares its authenticator α_b and sends $\{\alpha_b, \mathrm{IDS}_a, \mathrm{m}_a\}$ to tag T_b. Using its own key, T_b verifies α_b; then, computes its second authenticator β_b, partial evidence mb and sends them to the reader. Then, T_b updates its IDS_b. Upon receiving $\{\beta_b, \mathrm{mb}\}$, the reader verifies β_b and then concludes that T_a and T_b exist in the field, simultaneously. Finally, the reader accumulates $\{\mathrm{IDS}_a, \mathrm{IDS}_b, \mathrm{t}, \mathrm{m}_a, \mathrm{m}_b\}$ in a tuple, as a proof and, sends it to the back-end server. Notice that the reader does not update, at the end.

Wu uses a random permutation function F while calculating the authenticators and partial evidences. Wu claims F to be a one way function that uses only the PRNG and XOR operation available in a Gen-2 tag. The implementation of F function is shown by an example. Let $\mathrm{M} = (\mathrm{m}_0, \mathrm{m}_1, \mathrm{m}_2, \mathrm{m}_3)$, $\mathrm{C} = (\mathrm{c}_0, \mathrm{c}_1, \mathrm{c}_2, \mathrm{c}_3)$, $\mathrm{D} = (\mathrm{d}_0, \mathrm{d}_1, \mathrm{d}_2, \mathrm{d}_3)$, $\mathrm{E} = (\mathrm{e}_0, \mathrm{e}_1, \mathrm{e}_2, \mathrm{e}_3)$, where m_i, c_i, d_i, e_i and γ are all 16-bit numbers. Function $\gamma = \mathrm{P(E)} = \mathrm{PRNG(PRNG(PRNG(PRNG(e_0)\oplus e_1)\oplus e_2)\oplus e_3)}$. For $\mathrm{C} = \mathrm{F(M)}$; $\mathrm{c}_0 = \mathrm{P}(\mathrm{m}_0, \mathrm{m}_1, \mathrm{m}_2, \mathrm{m}_3)$, $\mathrm{c}_1 = \mathrm{P}(\mathrm{m}_1, \mathrm{m}_2, \mathrm{m}_3, \mathrm{m}_0)$, $\mathrm{c}_2 = \mathrm{P}(\mathrm{m}_2, \mathrm{m}_3, \mathrm{m}_0, \mathrm{m}_1)$, $\mathrm{c}_3 = \mathrm{P}(\mathrm{m}_3, \mathrm{m}_0, \mathrm{m}_1, \mathrm{m}_2)$. In brief, F(M) is a total of 16 nested

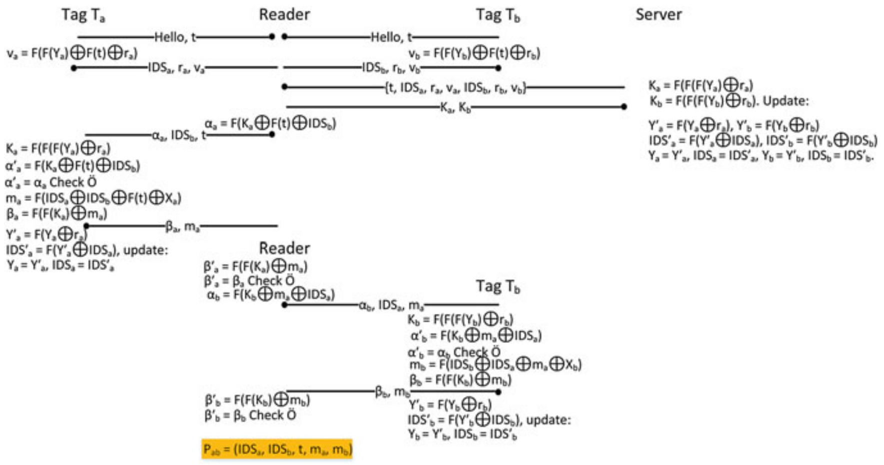


Fig. 3. The scheme of work [22]

PRNG and 12 XOR operations. Since F is a public function, for any known input, F(x) can be calculated. Therefore, a table similar to that of Table 1 in Sect. 3.1 can be prepared. Only the preparation of the table -called Table X- is computationally more intensive than the preparation of Table 1, but it will have 216 (65,536) possible inputs and corresponding outputs as Table 1. Referring to the input and output columns of Table X is no different than that of Table 1 and takes very short time. Additionally, F(F(x)) is the application of F function on the result of F(x), i.e. 32 PRNG and 24 XOR operations.

4.1 Attacks on Wu's Scheme

Exposure Attack. The exposure attack on the protocol is similar to the attack in Sect. 3.1. The adversary challenges the tags, with a bogus timestamp t. In the replies of tags, the IDS and nonce values are recorded, then the authenticators v$_a$ and v$_b$, are analyzed.

Referring to the hypothetical Table X, the value of v$_a$ is used as an output and the corresponding input -called input$_v a$- is read. From Fig. 3:

$$input_{va} = F(Y_a) \oplus F(t) \oplus r_a \tag{1}$$

$$F(Y_a) = input_{va} \oplus F(t) \oplus r_a \tag{2}$$

The nonces r$_a$, r$_b$, IDS$_a$, IDS$_b$ and the timestamp t are in clear text and the value of F(t) is found from the output column of Table X. Hence, by XORing the found input$_v a$ with the known values F(Y$_a$) is exposed (Eq. 2). Using the exposed F(Y$_a$) in the output column of Table X, the value in the input column gives Y$_a$. Following the same steps Y$_b$ is also exposed. The eys K$_a$, K$_b$ are calculated using the newly exposed Y_a and Y_b, the nonces r_a and r_b. Next, the updated values of Y_a and Y_b are also exposed. The updated values of IDS_a and IDS_b depend on the current values and the updated values of Y_a and Y_b, exposed in the previous step. Thus, the adversary also obtains the updated values of IDS_a, IDS_b. Without eavesdropping any message exchanges the adversary captures Y_a, Y_b, K_a, K_b and updated values of Y_a, Y_b, IDS_a, IDS_b.

The attack continues by eavesdropping one complete round. When the reader sends $\{\alpha_a$, IDS_b, t$\}$ to Ta, the attacker waits for the reply $\{\beta_a$, $m_a\}$. Using the value of m_a in the output column, a corresponding input – call it input m_a - is read:

$$input_{ma} = IDS_a \oplus IDS_b \oplus F(t) \oplus X_a \tag{3}$$

$$X_a = input_{ma} \oplus IDS_a \oplus IDS_b \oplus F(t) \tag{4}$$

Hence, the pre-shared secret X_a is captured (Eq. 4). The message $\{\alpha_b$, IDS_a, $m_a\}$ is of no importance because its terms are captured values. The reply $\{\beta_b$, $m_b\}$ of T_b gives away the pre-shared secret X_b, after a similar analysis of m_b, as in m_a. Hence, the adversary has the shared secrets X_a and X_b, necessary for the creation of rogue tags. The rest of the attack is the same as in Sect. 3.1. The attacker loads the captured values into two rogue tags, switching the identities of the target and conveyor. The result is wrong medication of a targeted inpatient, possibly causing deadly conditions.

De-synchronization Attack. The protocol of Fig. 3 is also vulnerable to de-synchronization attack, at many points. De-synchronization happens when one of the partners of the message exchange update some shared terms to new values, while the other does not. If the old values are not stored, then there is no way for mutual authentication to take place, with mismatched values. For example, consider the moment when the reader sends the messages $\{IDS_a$, ra, $v_a\}$ and $\{IDS_b$, rb, $v_b\}$, to the server. After calculating and sending the keys, the server updates. If the reader does not get the keys (loss of power), or cannot continue communication with the tags, then the server is de-synchronized with the tags; because the tags have not been updated. During the retry, the reader obtains and sends the old $\{IDS_a$, ra, $v_a\}$ and $\{IDS_b$, rb, $v_b\}$. The server never finds the old values in its database to verify the tags. In total, there are four instances that can cause de-synchronization: the reply of the server to the reader, the message exchange between the reader and T_a, the message of reader to T_b. Extra care is necessary in protocols that use updating, because de-synchronization halts medication.

4.2 Computational Load of Wu's Scheme on Gen-2 Tags

Looking at Fig. 3, the most intensive computations in tags take place, after receiving authenticator (α_a, α_b) of the reader. Counting the number of F function and XOR operations from the instant of computing Ka until sending $\{\beta_a$, $m_a\}$

(assuming update can take place after sending {β_a, m$_a$}); Ta has a larger computational load with nine F function and seven XOR operations. Every F function involves sixteen PRNG and twelve XOR operations. Hence, Ta makes 144 PRNG and 115 XOR operations. A PRNG consumes around 190 clock cycles to produce a random number [24]. Assuming a 16-bit architecture, each XOR operation takes one clock. In total, Ta spends 27,475 (144×190+115) clock cycles in computations. This is around 26 times more than an 8-bit AES implementation, which consumes 1032 clock cycles [25]. In other words, Wu's proposal cannot meet the limits, as it exceeds 220 clock cycles [26].

5 Discussions

As demonstrated, Yen's and Wu's schemes are as vulnerable as their predecessors, which contradict the major safety goals [2,4]. Wu's scheme cannot fit in a Gen-2 tag, is vulnerable and has the same characteristics of Wu's and Yen's protocols; therefore, it will not be discussed any further. The reason of the disclosure of critical data by our attacks is a result of using the only available function PRNG, as an encryption function. To the best of our knowledge there is no formal proof of using a PRNG as an encryption or hashing algorithm [19]. For patient safety, confidentiality of critical data has to be provided by true encryption. In other words, an alternative with stronger cryptographic primitives is necessary, instead of the 16-bit PRNG function of ISO 18000-6 or EPC Gen-2 tags, which are used for commercial goods in supply chains. Bit size of PRNG can be extended to 64 or more bits to increase the search space of the unknowns given as input to the PRNG which makes creating a table and searching through the table unaffordable. Even more, PRNG function can be replaced by a better cryptographic function. But these extensions mean to change the EPC Gen2 standard.

5.1 Ambiguities and Disadvantages of Yen's Proposal

Yen's proposal carries over the ambiguities of the work it criticizes. Even if PRNG is accepted as the only viable option, a special tag is required to calculate values like PRNG(ID$_{Pi}$⊕r$_b$⊕ K$_{Pi}$), because in regular EPC Gen 2 tags PRNG function doesn't have any input parameters. Another unexplained assumption is the digital signing ability of the tags. This assumption is highly questionable as the only available option is a PRNG and suggesting its use in digital signing is totally unacceptable. An unconsidered but possible scenario is the presence of more than one inpatient, in the same room. UHF tags are read in numbers from a few meters away. Thus, it is not possible to identify which inpatient's tag is read, if there are many in a room. With equal distance from two inpatients, a nurse can give the other patient's medicine to the intended patient. The aftermath of a complication at an inpatient is not considered, either. The medication responsibility of other inpatients, while a previous inpatient is going through a complication, is ambiguous. The continuation of medication with the same PDA, by a second nurse is not good. If a wrong medication is detected, the first nurse is falsely blamed. Using a second nurse PDA causes a discontinuation of the inpatient tuples and requires server intervention; since every nurse downloads inpatient data with her own PDA/password.

In their security analysis, Yen et al. claim that data confidentiality of their protocol is guaranteed. However, the identities and keys of the inpatients are exposed after our full disclosure attack, even though they are not transferred in plaintext.

Not only that, Yen's system has disadvantages, as well. A disadvantage is dedicating a PDA for every nurse, which is neither widely available nor cheap. UHF readers in the form of PDAs are uncommon and expensive. This increases the overall cost in hospitals, where there are many clinics and many shifts. Finally, the lack of consideration of Health Level 7 (HL7) standards is another disadvantage, because any incompliant solution is unlikely to be endorsed [27]. Not paying attention to HL7 standards is partly the reason of vulnerabilities; especially the mutual authentication requirement. Various attacks on ISO 18000-6 RFID are explained in detail [10,17,28]. In brief, there are four types of attacks: Interception, interruption, modification and fabrication attacks. Each attack has some counter measures, but they are not enough to guarantee patient safety, simply because of the limited resources of the tags, in question.

5.2 Security Vulnerabilities of Wu's Scheme

In their security evaluation, Wu et al. [22] defends that impersonation, ID-Theft and clone attacks cannot be launched against their protocol. Contrarily, our full disclosure attack exposes the secret keys, which opens the avenue to generating false grouping proof evidence. Not only that, our attack demonstrates how a fake tag (clone) can be devised to alter the identity of an impatient. Therefore, their clone attack evaluation is also unsatisfactory. Besides the successful impersonation and clone attacks, a de-synchronization attack is demonstrated above, an attack type they fail to evaluate in their analysis.

5.3 Suitable Technology for Patient Safety: NFC

A viable alternative technology is the near field communication (NFC) tags, because they possess the desired characteristics and cryptographic primitives. For example, Mifare DesFire version EV1 (EV1) tag has a built in AES engine [29]. If this feature existed in ISO 18000-6 tags, both of our PRNG table attacks would have been ineffective. Definitely, the existence of an AES engine provides better patient data safety. Another important characteristic that would have prevented our attacks is the operating distance. EV1 is read from a distance of 20–100 mm, therefore the nurse has to approach intentionally very close to an inpatient. Such a physical requirement removes the danger of eavesdropping by an adversary from meters away and the danger of reading a rogue tag.

The characteristics of EPC Gen-2 and EV1 tags that impact medicine administration are compared, in Table 2. Apart from the encryption and reading

Table 2. Comparison of EPC Gen-2 Tag and DesFire EV1

Property	EPC Gen-2 Tag	DesFire EV1
Authentication	$\oplus$	AES
Supply energy	No battery	No battery
Operating distance	Up to 7 m	20–100 mm
Tags read	1000 tags/s	1 tag/s
Data integrity	16 bit CRC, framing	16/32 bit CRC, parity, bit coding, bit counting
Memory capacity	512 bit on chip	2, 4, 8 kB NV-memory
Standard	ISO 18000-6	ISO/IEC 14443A

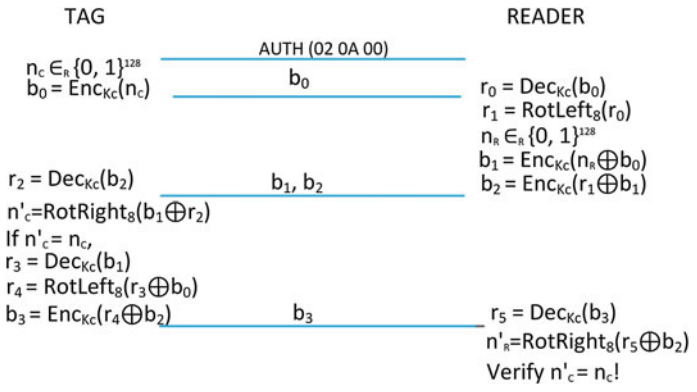

TAG
READER
AUTH (02 0A 00)
$n_c \in_R \{0,1\}^{128}$
$b_0 = Enc_{Kc}(n_c)$
b_0
$r_0 = Dec_{Kc}(b_0)$
$r_1 = RotLeft_8(r_0)$
$n_R \in_R \{0,1\}^{128}$
$b_1 = Enc_{Kc}(n_R \oplus b_0)$
$b_2 = Enc_{Kc}(r_1 \oplus b_1)$
$r_2 = Dec_{Kc}(b_2)$
b_1, b_2
$n'_c = RotRight_8(b_1 \oplus r_2)$
If $n'_c = n_c$,
$r_3 = Dec_{Kc}(b_1)$
$r_4 = RotLeft_8(r_3 \oplus b_0)$
$b_3 = Enc_{Kc}(r_4 \oplus b_2)$
b_3
$r_5 = Dec_{Kc}(b_3)$
$n'_R = RotRight_8(r_5 \oplus b_2)$
Verify $n'_c = n_c$!

Fig. 4. DesFire EV1 authentication

distance advantages, the NFC tags have other advantages over the EPC Gen-2 tags. Data integrity of the exchanged messages is an important security characteristic. Any multiple changes in the transmitted messages should be detected. As observed from the table, EV1 provides better data integrity algorithms. But, a property where EPC Gen-2 technology performs better is the number of tags read per second. A nurse can read only maximum one NFC tag/s, because physically she has to approach and momentarily touch the tag. But, this does not provide an advantage over NFC tags because; there is no hurry to read many inpatient tags. The memory capacity of EV1 tags surpasses the EPC Gen-2 tags. This is important because future protocols and schemes have a better chance to be accommodated on a spacious EV1 tag. User developed security applications or extended secrets can be stored in EV1. The ISO standards of the two technologies are different. But, the ISO 14443A standard is meant for the smartcards, clearly a higher class technology than the ISO 18000-6 standard.

Another important parameter of the EV1 is its 3-way mutual authentication. As given in Fig. 4, a new session key is created for each session through a pre-shared key. Simply, the authentication is based on the verification of the exchanged encrypted nonces.

The use of strong cryptographic primitives instead of a simple 16-bit PRNG function increases the security level but also leads to other hardware requirements. Therefore, one would expect to see a considerably more expensive cost for the higher NFC technology solution. But, this is not the case. The NFC tag prices are higher than the UHF tags, but the total cost for a complete solution is not. In his cost analysis, Peris et al. calculate a total cost for a floor with 5000 inpatients/year, 3 unit-dose/day, 3 nurses on each floor and an average hospital with 8 floors [18]. The cost of the HIS and AMD are excluded, because those are included in the overall cost of the hospital. The cost of an EPC Gen-2 tag is given as $0.5/tag, including the plastic package of each unit-dose. Every nurse is equipped with a PDA, astonishingly priced at $300. The total number of tags used for the inpatients and the unit doses is 20,000/year; mistakenly taken as 15,000/year by Peris et al. In the end, Peris et al. conclude with a cost of $70,000/year, for his proposal. An NFC tag costs $0.421 to $0.825, depending on the size of the order. Hence, the NFC tags are more expensive than EPC Gen-2 tags, as expected. But, to the best of our knowledge, a mobile UHF Gen-2 reader is around $1027. On the other hand, a popular NFC enabled tablet (Google Nexus 7) costs around $199. Therefore, there is a 5:1 price ratio, in favor of NFC readers. Obviously, even with the most expensive NFC tag, our solution ($21,300) is less expensive than that of Peris et al.'s.

6 Conclusion

The weaknesses of ISO-18000-6 or Gen-2 tags in safe drug administration are obvious, following the various attacks presented in this and previous work. As demonstrated the analyzed protocols fail their data confidentiality claims. The use of PRNG as an encryption algorithm is a major drawback. On the other hand, those proposals that try to provide stronger encryption by nested PRNG operations, cannot meet the time limits of RFID tags. With so many weaknesses and disadvantages, EPC Gen-2 type tags cannot increase inpatient medication safety.

There is a need for tags with cryptographic primitives, intentional tag reading characteristics and longer key sizes. State of the art NFC tags are a viable alternative. The previous works suggest the use of non-standard operations and special equipment. The contemporary, less expensive and widely available NFC enabled tablets are better suited for the job. The comparison of EPC Gen-2 and NFC tag technologies indicate that NFC is a better viability. Currently, a proposal using the NFC technology and strong security is underway, in our lab. An authentication based on EV1 mutual authentication structure will be the future work. Another alternative to increase the security is using public-key cryptography as indicated in [30], but the huge clock cycle (66,048) of using that alternative affects usability of the system.

References

1. Eurobarometer 2006 survey on medical errors. Technical Report, October 2005 (2006)
2. Kaushal, R., Bates, D.W., Landrigan, C., McKenna, K.J., Clapp, M.D., Federico, F., Goldmann, D.A.: Medication errors and adverse drug events in pediatric inpatients. JAMA **285**(16), 2114–2120 (2001)
3. Reporting, E., Meaders, S., Hickner, J., Kuo, G.M., Fagnan, L.J., Forjuoh, S.N., Stevens, B.K., Pace, W.D., Hamlin, B.N., Scherer, H., Hudson, B.L., Oppenheimer, C.C.: Field test results of a new ambulatory care medication error and adverse drug. Ann. Family Med. **8**, 517–525 (2010)
4. 2010 National Patient Safety Goals (NPSGs) (2010)
5. Wal-Mart details its RFID journey. http://www.computerworld.com/s/article/109132/Wal_Mart_details_its_RFID_journey
6. Information technology - Radio frequency identification for item management - Part 6: Parameters for air interface communications at 860 MHz to 960 MHz. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=46149
7. Prop, E.I.: EPC Global Class1 Gen2 RFID Specifications, December 2005 (2006)
8. Juels, A.: Yoking-proofs for RFID tags. In: Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, pp. 14–17, 138–143. IEEE Press, New York (2004)
9. Saito, J., Sakurai, K.: Grouping proof for RFID tags. In: Conference on Advanced Information Networking and Applications, pp. 621–624. Taichung (2005)
10. Peris-Lopez, P., Orfila, A., Hernandez-Castro, J.C., van der Lubbe, J.C.: Flaws on RFID grouping-proofs. Guidelines for future sound protocols. J. Netw. Comput. Appl. **34**(3), 833–845 (2011)
11. Wu, F., Kuo, F., Liu, L.W.: The application of RFID on drug safety of inpatient nursing healthcare. In: 7th International Conference on Electronic Commerce, pp. 85–92, New York (2005)
12. Sun, P.R., Wang, B.H., Wu, F.: A new method to guard inpatient medication safety by the implementation of RFID. J. Med. Sys. **32**, 327–332 (2008)
13. Chen, C.-L., Wu, C.-Y.: Using RFID yoking proof protocol to enhance inpatient medication safety. J. Med. Sys. **36**, 2849–2864 (2012)
14. Huang, H.H., Ku, C.Y.: A RFID grouping proof protocol for medication safety of inpatient. J. Med. Sys. **33**(6), 467–474 (2008)
15. Chien, H.Y., Yang, C.C., Wu, T.C., Lee, C.F.: Two RFID-based solutions to enhance inpatient medication safety. J. Med. Sys. **35**(3), 369–375 (2011)
16. Yen, Y.C., Lo, N.W., Wu, T.C.: Two RFID-based solutions for secure inpatient medication administration. J. Med. Sys. **36**(5), 2769–2778 (2012)
17. van Deursen, T., Radomirović, S.: Algebraic attacks on RFID protocols. In: Markowitch, O., Bilas, A., Hoepman, J.-H., Mitchell, C.J., Quisquater, J.-J. (eds.) WISTP 2009. LNCS, vol. 5746, pp. 38–51. Springer, Heidelberg (2009)
18. Peris-Lopez, P., Orfila, A., Mitrokotsa, A., van der Lubbe, J.C.A.: A comprehensive RFID solution to enhance inpatient medication safety. Int. J. Med. Inform. **80**, 13–24 (2011)
19. Wickboldt, A.K., Piramuthu, S.: Patient safety through RFID: vulnerabilities in recently proposed grouping protocols. J. Med. Sys. **36**(2), 431–435 (2012)
20. Kasper, T., von Maurich, I., Oswald, D., Paar, C.: Chameleon: a versatile emulator for contactless smartcards. In: Rhee, K.-H., Nyang, D. (eds.) ICISC 2010. LNCS, vol. 6829, pp. 189–206. Springer, Heidelberg (2011)

21. Redemske, R., Fletcher, R.: Design of UHF RFID emulators with applications to RFID testing and data transport. In: Fourth IEEE Workshop on Automatic Identification Advanced Technologies (AutoID 2005), pp. 193–198 (2005)
22. Wu, S., Chen, K., Zhu, Y.: A secure lightweight RFID binding proof protocol for medication errors and patient safety. J. Med. Sys. **36**(5), 2743–2749 (2012)
23. Yu, Y.C., Hou, T.W., Chiang, T.C.: Low cost RFID real lightweight binding proof protocol for medication errors and patient safety. J. Med. Sys. **36**(2), 823–828 (2012)
24. Peris-Lopez, P., Hernandez-Castro, J.C., Estevez-Tapiador, J.M., Ribagorda, A.: LAMED—A PRNG for EPC Class-1 Generation-2 RFID specification. Comput. Stand. Interfaces **31**(1), 88–97 (2009)
25. Feldhofer, M., Wolkerstorfer, J.: Hardware implementation of symmetric algorithms for RFID security. In: Kitsos, P., Zhang, Y. (eds.) RFID Security, pp. 373–415. Springer, New York (2009)
26. Martín, H., Millán, E.S., Entrena, L., César, J., Castro, H.: AKARI-X: a pseudorandom number generator for secure lightweight systems. In: 17th IEEE IOLTS, pp. 228–233 (2011)
27. Health Level Seven International. http://www.hl7.org/implement/standards/index.cfm?ref=nav
28. Hawrylak, P.J., Schimke, N., Hale, J., Papa, M.: Security risks associated with radio frequency identification in medical environments. J. Med. Sys. **536**(6), 3491–3505 (2012)
29. Mifare DesFire EV1 Leaflet. http://www.nxp.com/products/identification_and_security/smart_card_ics/mifare_smart_card_ics/mifare_desfire/
30. Oren, Y., Box, P.O.: A low-resource public-key identification scheme for RFID tags and sensor nodes a brief description of the protocol. In: WiSec 2009, pp. 59–68 (2009)

Rights Management with NFC Smartphones and Electronic ID Cards: A Proof of Concept for Modern Car Sharing

Timo Kasper(✉), Alexander Kühn, David Oswald, Christian Zenger, and Christof Paar

Horst Görtz Institute for IT Security, Ruhr-University Bochum, Bochum, Germany
{Timo.Kasper, Alexander.Kuehn, David.Oswald, Christian.Zenger, Christof.Paar}@rub.de

Abstract. Numerous contactless smartcards (and the corresponding RFID readers) are compatible with NFC, e.g., Mifare cards and the governmental ID card in Germany called nPA. NFC-enabled smartphones and other NFC objects such as door locks have become widespread. Existing and future applications of the up-and-coming technology require a secure way of assigning and transporting user rights, e.g., for opening and starting a car or access control to a building. In this paper, we propose a scheme that securely identifies a customer on a website and creates a (personalized) credential containing the booked access permissions. This credential is safely transported via the Internet to the user's smartphone and finally grants access to an NFC-enabled object. In our proof-of-concept implementation, an application on a commercial smartphone is used for communicating with a web server of a car rental agency. During the booking process, the phone operates as an RFID reader to interrogate the nPA of the user and utilizes the security mechanisms of the nPA, including the PACE protocol, for identifying the customer. After having obtained the credential, the smartphone emulates a Mifare DESFire card that is read by the NFC door lock of a rental car to verify the validity of the access permission. We discuss security issues and limitations of our approach.

Keywords: German electronic identity card · User rights management · Car sharing · Smartphone · NFC · Contactless smartcard emulation

1 Motivation

The ISO 14443 standard for contactless smartcards [28] was published around 2000. Today, many applications for ticketing, micro payments, access control, and identification rely on contactless cards that are compliant to this standard, e.g., NXP's Mifare family of cards, electronic IDentificationID cards, and passports. In these applications, the roles of the actively powered reader interrogating

This work was supported in part by the German Federal Ministry of Economics and Technology (Grant 01ME12025 SecMobil).

passive cards are clearly defined. A few years later, Near Field Communication (NFC) [29] emerged from ISO 14443 and is today implemented in many embedded devices, such as smartphones, door locks, and other objects. NFC is compatible to ISO 14443, but enables additional features: For instance, an NFC-enabled object can choose to appear as a reader in one moment and switch its role to appear as a contactless card in the next moment. Furthermore, an operating mode with an extended range is defined, whereas both participants function as active readers communicating with each other.

Major manufacturers of smartphones, e.g., Nokia, HTC, LG, and Samsung, offer NFC-enabled smartphones. The NFC-link is designed to operate at a distance of only a few centimeters and provides an additional communication interface to the ones already provided by the smartphone, e.g., Wireless Local Area Network (WLAN), Universal Mobile Telecommunications System (UMTS), Global System for Mobile Communications (GSM), Global Positioning System (GPS), or Bluetooth. Furthermore, the main processor of the smartphone can perform complex computations and can be employed to interconnect the communication interfaces of the smartphone.

Due to the compatibility with existing ISO 14443 Radio Frequency IDentification (RFID) readers and infrastructure, various other objects are already equipped with NFC technology, e.g., locks for hotel rooms and safe deposits, payment terminals, and ticketing solutions. Likewise, electronic passports and governmental ID cards, e.g., in Germany, as well as future electronic driver's licenses can be accessed with NFC. Recently, even cars — often already augmented with GPS receivers and a GSM or UMTS modem — additionally provide an NFC-enabled door lock. Thus, an excellent basis is given for realizing various new applications that combine services on the Internet with contactless cards and other NFC objects.

Outsourcing business and administrative tasks to the Internet potentially provides a significant gain regarding costs for companies. Therefore, the verification of a user's identity without a face-to-face meeting becomes more and more important. Modern car sharing solutions are emerging, e.g., in Germany DriveNow by BMW, Car2Go by Daimler, and Flinkster by Deutsche Bahn. Fleet management of vehicles, access to a safe deposit or post box, and opening doors of previously booked hotel rooms are some more possible business options that can be realized by means of NFC-enabled smartphones with Internet access: The issuer transfers an electronic key (or another right) via the Internet to the smartphone. Once stored on the smartphone, the electronic key can repeatedly be used to access a particular secured object via NFC.

The required components such as NFC-enabled locks and smartphones are available. However, for example for billing purposes, the identity of the customer must be verified securely. This is where governmental electronic IDs come into play, since they usually provide methods for an NFC-based, remote identification of the owner via the Internet. In this context, the protection of the individual's identity is of particular relevance. The new German electronic identity card (nPA) is designed to provide the required features, i.e., a mechanism for identification, authentication, and the protection of stored personal data, e.g., the identity of the owner. With the nPA, it is possible to identify an individual in passport controls, at airports, at governmental authorities, and on the Internet.

1.1 Contribution and Outline

Due to the complexity of the involved communication interfaces and technologies, for some system designers it is unclear how to address the security risks of NFC and how to realize the corresponding schemes for identifying the customer and transporting the booked rights from the issuer to the target object. Due to lack of publicly disclosed realizations of the mentioned applications, it is often uncertain, if according systems are practically feasible at all.

In this paper we aim to fill in this gap. To this end, in Sect. 3, we propose a scheme for (i) identifying a customer by combining the NFC interface and the Internet connection of the smartphone with the security capabilities of an electronic ID card, (ii) issuing a credential containing the rights assigned by the issuer to the customer and securely transferring the credential to the smartphone, (iii) assuring that the rights specified in the credential can be verified by the target object, e.g., an NFC-enabled door lock of a car, even in a scenario without a permanent Internet connection, and finally (iv) securely exchanging the credential via an NFC connection between the smartphone and the target object and decide whether to grant or deny access. As an alternative scenario for using the NFC phone, users shall be able to obtain user rights and execute granted rights on the basis of a standard contactless smartcard, i.e., compatibility to commercially available contacless cards shall be maintained, where possible.

As the main contribution, in Sect. 4, we illustrate the practical feasibility of the proposed scheme by realizing all relevant parts of the scheme as a proof-of-concept on a commercial smartphone and validating our implementation. We practically demonstrate that the smartphone is suitable for using the electronic identity (eID) function of modern governmental documents and realize the corresponding cryptographic schemes in Sect. 4.3, including the required computations on specific elliptic curves. We further implement the 3DES-based authentication scheme of Mifare DESFire cards in Sect. 4.4, and show that emulating a DESFire card with a commercial NFC smartphone is practical. In Sect. 4.5, we pinpoint implementation obstacles and finally discuss security issues in Sect. 5. In the following Sect. 2, the relevant fundamentals of the nPA, NFC, and Secure Element (SE) are described.

2 Fundamentals

In this section, we present the eID service of the nPA that provides a method for the online identification of customers. For the secure transmission of personal data, the Password Authenticated Connection Establishment (PACE) and Extended Access Control (EAC) security protocols of the nPA are described. Furthermore, important aspects of SE and NFC technology are shown.

2.1 The German Electronic Identity Card (nPA)

With the introduction of the nPA in Germany on November 1st, 2010, a new system for the administration of electronic identities has been established. The so-called eID service is a service supported by the nPA for verifying the correctness of an electronic identity on the Internet. For using the secure authentication on the Internet, e.g., to open a bank account, for online shopping, or administrative tasks, an nPA, and a Personal Computer (PC) with an RFID reader [15] and Internet access are required. For verifying the owner's identity, a multi-factor authentication based on *knowing* a Personal Identification Number (PIN) and *owning* the nPA is employed [16].

eID Service. The eID service [14] of the nPA enables to verify the identity of the passport owner. As a trust anchor, an eID server, e.g., operated by Bundesdruckerei GmbH in Germany, participates in the identification process. The required personal customer data, stored on the nPA and signed by the Federal Ministry of the Interior of the German Federal Republic, is redirected to the service provider by the eID service. For securing the personal data, various cryptographic mechanisms, e.g., Advanced Encryption Standard (AES), Elliptic Curve Cryptography (ECC), RSA, and SHA-1 are implemented on the nPA.

The communication between an nPA, an RFID reader, and the participating eID server is secured by the PACE protocol and the EAC protocol. For validating the authenticity of the personal nPA data, its signature has to be verified with the corresponding public key. The authenticity of this public key can in turn be verified with a certificate issued by a Root Certificate Authority (RCA) belonging to the Public Key Infrastructure (PKI) [7] of the Federal Office for Information Security [16].

Password Authenticated Connection Establishment. The PACE protocol is the base security protocol of the nPA. It establishes a secured communication channel between the nPA and the RFID reader using the Diffie-Hellman Key Exchange (DHKE) or the Elliptic Curve Diffie-Hellman Key Exchange (ECDHKE). For a successful execution of the protocol, a PIN, e.g., the number printed on the ID card or an individual, secret eID key only known to the nPA holder, has to be entered.

During the PACE protocol, two shared session keys are generated, namely an encryption key K_{enc} and a key K_{MAC} for authentication with a Cipher-based MAC (CMAC) [12]. After a successful execution of the PACE protocol, mutual authentication between RFID reader and nPA is established. More information about the PACE protocol can be found in [17,27].

Extended Access Control. The EAC protocol relies on a successful PACE execution and uses the generated key pairs. EAC consists of two parts, the Terminal Authentication (TA) and the Chip Authentication (CA). The TA uses a challenge-response protocol for enabling the nPA to verify the authentication of the terminal and to define the terminal's access permissions to the personal data. In the CA, the nPA proves its authenticity and the correctness of the stored data to the terminal. While the PACE protocol establishes a secure communication channel between the nPA and the RFID reader, the EAC protocol secures the communication from end to end between the nPA and a remote server. The Secure Messaging (SM) of the EAC employs newly generated session keys of the current protocol run, as further detailed in [17,18].

2.2 Operating Modes of Near Field Communication

NFC is a wireless communication technology, defined in the ISO/IEC 18092 standard [29] and is compatible to ISO/IEC 14443, i.e., based on RFID communication at a frequency of 13.56 MHz. The standard specifies three modes of operation in which NFC can be communicate:

Reader mode (active) An NFC-enabled device in active mode (Proximity Coupling Device (PCD)) can access a passive NFC tag, e.g., a contactless card (Proximity Integrated Circuit Card (PICC)) or an NFC device in passive mode.

Card emulation mode (passive) An NFC-enabled device in passive mode emulates a contactless smartcard and thus acts as a PICC.

Peer-to-peer mode Two NFC-enabled reader devices both operate in active mode (as PCDs), i.e., actively transmit their data.

The first two operation modes are used in our concept and its implementation: The NFC smartphone acts in the reader mode for the communication with the nPA, while the card emulation mode is used for the communication with the NFC-enabled door lock of the car.

2.3 Secure Elements

An attacker can gain unauthorized access to the Operating System (OS), e.g., by means of a Trojan horse, and get hold of secret data. Thus, all data held in the OS of the smartphone and all computations carried out by the main processor can be possibly read out by an attacker using malware. Thus, for securely storing (small amounts of) security-sensitive data and executing security-related algorithms with low computational demands an SE should be used: Besides tamper-proof volatile and non-volatile memories, it contains a secure microprocessor (often with an 8-bit architecture) that usually runs a Javacard OS. Three distinct types of SEs are common in NFC smartphones:

1. an embedded Secure Element (eSE) that is realized by the manufacturer of the smartphone, either as a separate chip connected to the NFC circuitry or directly integrated into the NFC chip itself
2. a Subscriber Identity Module (SIM) card issued by the mobile communications provider
3. an SE embedded into an (micro)SD card can be used, if the smartphone provides a corresponding slot

For using the SE, access must be granted to the developer, which often poses a problem in practice. Maybe, trusted zones in main processors, such as the ARM TrustZone [41], constitute an alternative to SE: TrustZone is a software-based security solution that is used in mobile devices with integrated ARM processor for securing the access to confidential data and secure online transactions.

3 Concept of Right Management

In this section, our concept for the secure management of user rights for NFC applications is presented in the context of a car sharing scenario. After introducing the basic idea, the involved participants and the concept goals, the three major parts of our concept, registration, credential request, and credential usage, are illustrated in detail.

3.1 Introduction to Mobile Rights Management

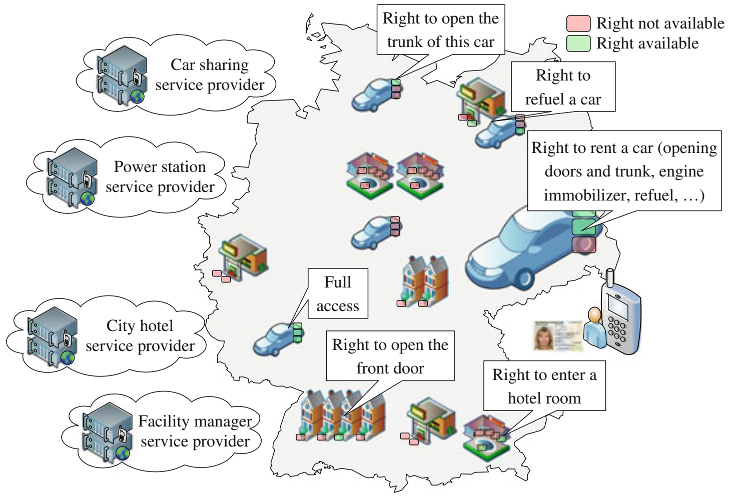

Right to open the trunk of this car
🟥 Right not available
🟩 Right available
Car sharing service provider
Right to refuel a car
Right to rent a car (opening doors and trunk, engine immobilizer, refuel, …)
Power station service provider
Full access
City hotel service provider
Right to open the front door
Right to enter a hotel room
Facility manager service provider

Fig. 1. Examples for actions and corresponding rights in our proposed scheme

In the first quarter of 2013, 210 million smartphones were sold worldwide [22]. The more than one billion smartphones currently in use can thus serve as an economical, flexible, and mobile alternative to existing infrastructures. A key advantage of NFC-enabled smartphones is that the mobile service provider does not have to hand out additional (electronic) tokens, but can instead provide a software-application that can be downloaded to the user's phone.

We present our concept for the secure managing of user rights exemplarily for mobile car sharing. As shown in Fig. 1, our approach is not limited to this single use-case, but can rather be applied to various other applications in which a smartphone user is required to securely obtain credentials. In our scheme, a credential can contain any kind of rights, e.g., the right to access a building or to refill the battery of an electronic vehicle at a charging station.

3.2 Participants

Our proposed concept distinguishes three participants. *Customers* want to use mobile services. *Service providers* are — besides providing the services — responsible for the distribution of different permissions for using the services, i.e., manage the credentials. The *trusted eID service*, e.g., the Bundesdruckerei GmbH, ensures the identity of the customer to the service providers.

The customer possesses an NFC-enabled smartphone with Internet access and an nPA. She knows the secret PIN for the eID application of the nPA. She wants to flexibly use different services on the go, for example, renting a car or booking a hotel room without face-to-face meetings with the vendor.

The service provider offers a smartphone application that serves as a reservation system and gives access to other service-related information to the customer. Further, the application contains the public keys of the service provider that can be used to secure the communication. The service provider operates an Internet server for the issuance and administration of credentials. Each customer who wants to use one of the services, e.g., rent a specific car at a specific time, has to send a request via the smartphone application to obtain a valid credential for this service. Thus, the task of the service provider is to administrate the generation of service permissions on request and securely distribute them to the correct customer's smartphone.

3.3 Registration

Before using a car sharing service, customers have to be registered once for billing purposes and to ensure that the customer possesses a valid driver's license. For that reason, in current realistic scenarios the customer has to visit one of the service provider's shops and present her driver's license.[1]

For the registration, the customer has to enter a secure, confidential password that is used for the generation of an individual customer public key pair, consisting of a public key pk_C, a secret key sk_C, and a certificate $cert_{pkC}$ for the public key of the customer pk_C. This data is transferred to the SE. The public keys pk_C of all customers are stored in a data base of the service provider in combination with the Machine Readable Zone (MRZ) of the customer's nPA for binding the public key pair to the nPA. Furthermore, the smartphone stores pk_{SP}.

Likewise, each NFC service object securely creates and stores an individual public key pair (pk_{SO}, sk_{SO}), the public key of the service provider pk_{SP}, and

[1] In future scenarios in which an electronic driver's license is available, this visit could probably be omitted

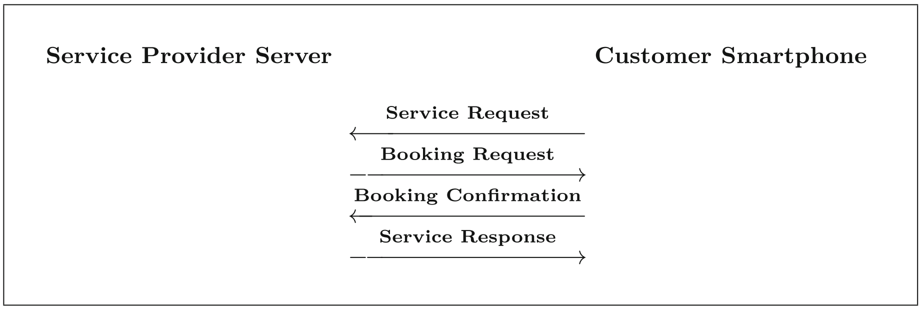


Fig. 2. Exchanged messages between service provider and smartphone

unique service object information. The latter is also stored, together with pk_{SO}, in the database of the service provider. Each service object and the service provider know and can securely execute a key derivation function, using an *Access Variable*, the service object information, and the public key pk_{SO} as inputs, for generating an *Authentication Key* (see Sect. 3.5).

3.4 Booking a Right

For obtaining the right for a particular service via the Internet, a request has to be issued to the service provider. Afterwards, the corresponding credential needs to be created and transferred to the smartphone of the customer. All communication via the Internet is secured with the Transport Layer Security (TLS) protocol [11]. The booking process consists of four main steps, as illustrated in Fig. 2.

Service Request. If the customer wants to use a service, e.g., book a rental car as close as possible to her current position, she starts the application on her smartphone that sends a request containing service-relevant information I_{SReq}, e.g., the GPS coordinates of the smartphone and the desired time of departure, to the service provider. In addition to the service information, the request contains the customer certificate $cert_{pkC}$. A random nonce N_C and a timestamp ts_{SReq}, both generated on the SE, are added as security values.

The service information I_{SReq} and the security values are concatenated and hashed by a secure hash function, e.g., SHA-512 [2,13]. Afterwards, the cryptographic security token t_{SReq} is generated signing the hash value with the customer's private key sk_C on the SE of the smartphone. The nonce N_C is stored on the SE. The service information, the security values, the the customer certificate $cert_{pkC}$, and the cryptographic security token are concatenated and encrypted with the public key pk_{SP} of the service provider. The encrypted data packet p_{SReq} is sent as the service request to the service provider. In summary, the following computations are executed:

$$\begin{aligned} h_{SReq} &:= \text{hash}(I_{SReq} \,||\, N_C || \, ts_{SReq}) \\ t_{SReq} &:= \text{sign}_{skC}(h_{SReq}) \\ p_{SReq} &:= \text{encrypt}_{pkSP}(I_{SReq} \,||\, N_C \,||\, ts_{SReq} \,||\, cert_{pkC} \,||\, t_{SReq}) \end{aligned}$$

After receiving the service request, the service provider decrypts it with his secret key sk_{SP}. The authenticity is ensured by means of the security values and the cryptographic token: The signature is verified with the customer's public key pk_C that authenticity can be validated using the certificate $cert_{pkC}$. The integrity of the received data packet is checked by computing the hash value of the received service information and security values and comparing it with the received hash value. The freshness of the received service request is guaranteed by means of the nonce and the timestamp ts_{SReq}. For the time verification by verifying the timestamps and the execution of user rights at the granted time period, the clocks of all participants have to be synchronous. If an error occurs, e.g., because the timestamp is out of time, a signed and encrypted error message is sent to the customer smartphone and the service request is rejected.

Identity Verification. The service provider uses the eID functionality of the nPA, described in Sect. 2.1, for reading out data of the customer's nPA and verifying the customer's identity, as illustrated in Fig. 3.

The NFC communication between nPA and smartphone is secured by means of the PACE protocol, during which the eID PIN — only known by the customer — has to be entered. The communication between smartphone, trusted eID service, and the service provider is secured by the TLS protocol. If the identity

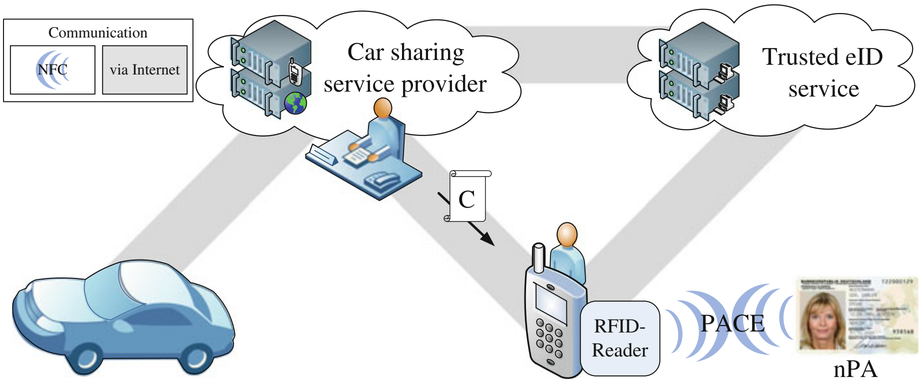


Fig. 3. Using the eID function of an electronic identity card (nPA) for booking a car and obtaining a credential C containing the assigned right: The smartphone acts as an RFID reader and communicates with the nPA using the PACE protocol. The authenticity of the nPA and the identity of the customer is then securely checked via Internet with the help of a trusted eID service and forwarded to the car sharing service provider. Furthermore, certain security parameters of the cars in the field can updated on a regular basis, e.g., via GSM or UMTS.

verification fails, a signed and encrypted error message is sent to the customer smartphone and the service request is rejected.

Service Credential Generation and Service Response. After a successful authentication and identification of the customer, the service provider modifies the nonce N_C to $N_C{}'$ in a defined way, which is also known to the SE in the smartphone. Afterwards, a booking request is generated by concatenating the service information I_{BReq} (e.g., coordinates of the car and the *User Right Validity Period*), the unique service object information UI_{BReq} (e.g., an unique car ID), the modified nonce $N_C{}'$, and the timestamp ts_{BReq}. This data is hashed and the hash value is signed with the private key sk_{SP} of the service provider. The data and the signed hash value are again concatenated with the customer certificate $cert_{pkC}$, encrypted with the customer's public key pk_C, and sent as a booking request p_{BReq} to the smartphone of the customer over a TLS-secured communication channel.

$$\begin{aligned}
h_{BReq} &:= \text{hash}(I_{BReq} \mid\mid UI_{BReq} \mid\mid N_C{}' \mid\mid ts_{BReq}) \\
t_{BReq} &:= \text{sign}_{skSP}(h_{BReq}) \\
p_{BReq} &:= encrypt_{pkC}(I_{BReq} \mid\mid UI_{BReq} \mid\mid N_C{}' \mid\mid ts_{BReq} \mid\mid cert_{pkC} \mid\mid t_{BReq})
\end{aligned}$$

The customer smartphone decrypts the received booking request on the SE using the stored sk_C. Then, the signature and the hash value are verified. The received modified nonce $N_C{}'$ is re-calculated on the basis of the original nonce N_C stored on the SE to ensure that the booking request of the provider belongs to the corresponding service request of the smartphone. The contained timestamp proves the freshness of the booking request. In case of an error, a signed and encrypted error message is sent to the service provider and the booking request of the service provider will be sent a second time. If the second try also fails, the current service request is canceled and a new service request has to be sent.

If the verification of data authenticity and integrity was successful, the received service information is displayed to the customer and in case of correctness and customer acceptance, the customer finally confirms the displayed booking offer by sending of a booking confirmation to the service provider. The SE of the customer smartphone modifies the received modified nonce $N_C{}'$, again in a pre-defined way known to both the phone and the provider to obtain $N_C{}''$. Then, the booking confirmation is generated with the service information I_{BReq}, the unique service object information UI_{BReq}, the modified nonce $N_C{}''$ and the timestamp ts_{BCon} that is hashed, signed, encrypted, and sent back to the service provider.

$$\begin{aligned}
h_{BCon} &:= \text{hash}(I_{BReq} \mid\mid UI_{BReq} \mid\mid N_C{}'' \mid\mid ts_{BCon}) \\
t_{BCon} &:= \text{sign}_{skC(h_{BCon})} \\
p_{BCon} &:= \text{encrypt}_p k_{SP}(I_{BReq} \mid\mid UI_{BReq} \mid\mid N_C{}'' \mid\mid ts_{BCon} \mid\mid t_{BCon})
\end{aligned}$$

The service provider decrypts the received data and verifies the correctness of the signature, the hash value, and the timestamp. The nonce $N_C{}''$ is checked by means of $N_C{}'$ known to the provider. If the received nonce is correct, the service provider can conclude that the service credential has been successfully decrypted with the private key sk_C of the customer and that the booking confirmation belongs to the previously transferred response.

If the service information I_{BReq} and the unique service object information UI_{BReq} of the booking confirmation matches the data of the service provider response, the service provider generates the *Service Credential*. It contains the service information I_{SC}, the unique service object information UI_{SC}, an *Authentication Key* used for securing the communication with the NFC object, e.g., the car's door lock, the encrypted *User Rights Credential*, the again modified nonce, and the timestamp ts_{SC}. The *Authentication Key* is generated using the key derivation function described in Sect. 3.3. The *Service Credential* is hashed, signed, encrypted, and sent to the customer smartphone.

$$\begin{aligned} h_{SC} &:= \text{hash}(Service\,Credential) \\ t_{SC} &:= \text{sign}_{skSP}(hSC) \\ p_{SC} &:= \text{encrypt}_{pkC}(Service\,Credential \mid\mid t_{SC}) \end{aligned}$$

After sending out the *Service Credential* as service response, the online ticketing service is successfully completed and the service provider can issue a bill to the correct customer. On the side of the smartphone, the received service response with the *Service Credential* is decrypted, its signature and hash value are verified, and the credential is stored in the SE of the card.

3.5 Executing the Granted Rights

For executing the obtained right at the booked NFC-enabled service object, the corresponding *User Rights Credential* — stored in the SE of the customer — needs to be securely transferred via NFC. For this purpose, the smartphone emulates a contactless smartcard with a suitable mutual authentication protocol for opening an NFC-enabled car, as illustrated in Fig. 4. In our realization of the scheme, a Mifare DESFire card [35] is exemplarily emulated,[2] however, other cryptographic contactless smartcards could serve for the same task.

For the emulation of a Mifare DESFire card, the NFC smartphone uses the 112-bit *Authentication Key* of the NFC object (contained in the *Service Credential*) for the mutual authentication and for establishing a secure channel with 3DES according to the proprietary protocol [32]. From the point of view of the NFC object, e.g., the car to be opened, the encrypted *User Rights Credential* appears to be stored in a data file of the DESFire card.

If the RFID reader of an NFC object detects an emulated contactless smartcard (or a real contactless smartcard) in its vicinity, the NFC object initiates the communication: It generates the required *Authentication Key* by means of the key derivation function detailed in Sect. 3.3. Next, the *Authenticate* procedure of the Mifare DESFire smartcard is executed between the smartphone and

[2] We opted to emulate a real-world card, because this enables additional use cases in which a real contactless card executes a (pre-paid) right instead of the smartphone.

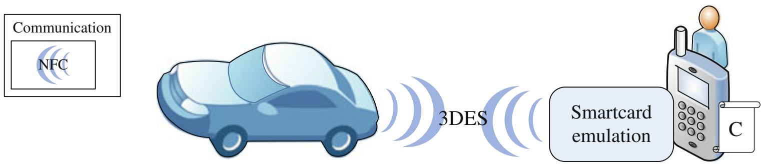


Fig. 4. For executing the granted rights at an NFC object, the smartphone emulates a contactless cryptographic smartcard (here, a Mifare DESFire card). The NFC reader in the door lock of the car detects the presence of the smartphone and initiates the communication. After establishing a secure channel (with 3DES), the credential C is transferred from the phone to the car.

the NFC object. As an outcome of the mutual authentication, smartphone and object agree on a *Session Key* for enciphering the current communication session with 3DES in Cipher Block Chaining (CBC) mode.

The service object reads out the encrypted *User Rights Credential*, protected from eavesdropping attacks, decrypts it with the private key sk_{SO}, and verifies the contained information by means of the signature of the service provider. If the unique service object information of the received *User Rights Credential* matches the service object, the access rights are granted and the service object can be used by the customer. For example, a car can be opened, and/or its engine can be started during a defined time slot. This service activation can be repeatedly executed as defined in the car sharing information, e.g., during the booking period.

3.6 Updating the Service Objects

Each service object securely and confidentially stores an own intern variable, the *Access Variable*, on the SE that is used for the derivation of the *Authentication Key*. The *Authentication Key* is newly generated in the SE after each *User Right Validity Period* using a service object-specific key derivation function with the *Access Variable* and the unique service object information as input.

After each *User Right Validity Period* on service object the service provider tries to securely exchange the confidential *Access Variable* via Internet. In case of missing Internet connection, the service object changes the *Access Variable* in a predefined way, also known to the service provider. The *Access Variable* is again modified after each completed *User Right Validity Period* and used for the calculation of the next *Authentication Keys* until the service object has Internet access for exchanging the intern *Access Variable* at a specified time.

The *User Right Validity Period* is specified as completed, if the authentication between service object and (emulated) smartcard was successful, the *User Right Credential* could be read out and was correct and the *User Right Validity Period* ran out.

The next time the service object has Internet connectivity, the *Access Variable* can be securely exchanged in the SE of the service object. For the refresh of the *Access Variable* the service provider sends a message to the service object encrypted with the public key pk_{SO} of the service object. The message contains the new *Access Variable*, the time to replace the old *Access Variable*, a nonce N_{SP} and a timestamp ts_{SP}. This data is hashed and the hash value is signed by the service provider. The use of the service object public key pk_{SO} binds the *Access Variable* to the one service object. After receiving the message, the service object exchanges the *Access Variable* at the specified time. The service object modifies the nonce N_{SP} in a predefined way, also known to the service provider, to $N_{SP}{}'$ and generates a response message containing the changed nonce $N_{SP}{}'$ and a timestamp ts_{SO}, hashed and signed with the service object private key sk_{SO}. The entire data is encrypted with pk_{SP} and sent back to the service provider. This way is ensured that the message for exchanging the *Access Variable* has reached the SE of the service object.

The important point of exchanging the *Access Variable* only at the specified time is founded in the requirement that already booked services for the service object have to remain valid after exchanging the *Access Variable*. If a *User Right Validity Period* is too far in the future that a variable exchange would be too late related to the security, the service provider can send a new *Service Credential* with a new *Authentication Key* to this customer.

4 Proof-of-Concept Implementation

We have fully realized a working Java implementation of our concept on an NFC smartphone in a lab setup and verified the functionality with different real nPA. Due to lack of a car with an NFC-enabled door lock, a PC connected to a standard NFC reader serves as a replacement for the car. Our realization is implemented with the BlackBerry Eclipse Java plug-in 1.5.2 for Eclipse version 3.7.0 (Eclipse Indigo) that supports *JRE 6 System Libary* offering many NFC functions and cryptographic classes, but on the other hand lacks support for the nPA Elliptic Curve (EC); this problem is further treated in Sect. 4.2. In the following, we present the target platform and selected parts of the implementation with a focus on the time-critical operations.

4.1 NFC Smartphone Blackberry Bold 9900

As a target platform for our demonstrator, we opted for the NFC-enabled smartphone Blackberry Bold 9900 [4] produced by Research In Motion (RIM). It provides a Qualcomm MSM8655 processor with an ARMv7 instruction set running at 1.2 GHz and 768 MB RAM. The SECUREAD NFC Solution Module by Inside Secure [26] realizes the NFC hardware of the phone. The usage of the smartphone's NFC capabilities is well documented, e.g., in [5,37,38]. In addition to NFC, the phone supports various other wireless communication methods, such as UMTS, WLAN, GSM, Bluetooth, and GPS. The phone furthermore offers a slot for microSD cards that might be used as an alternative to the built-in eSE of the NFC interface and the SIM card.

The OS of the Bold 9900 natively supports the passive NFC mode, i.e., emulating of a contactless smartcard. This is a key advantage in comparison to current Android-based smartphones, in which the original OS has to be modified, e.g., by means of CyanogenMod [10], in order to emulate a contactless smartcard [1].

4.2 ECC for PACE

The PACE protocol for communicating with the nPA requires an ECC implementation on the smartphone. Even though ECC is the most efficient established public-key scheme, it is still computationally intensive, and thus a bottleneck of the performance of our concept.

The elliptic curve used by the nPA is based on a 256-bit prime field $\mathbb{Z}_p$. The modulus of $\mathbb{Z}_p$ is a generalized Mersenne prime which allows for an efficient implementation of the reduction, and thus enables an efficient implementation of curve arithmetic. Unfortunately, the Application Programming Interface (API) provided by RIM does not support arithmetics on the required elliptic curve *brainpoolP256r1* [33], hence, we had to implement our own ECC arithmetics.

Point multiplication using affine coordinates, analogous to the exponentiation in multiplicative groups, turned out to be the most efficient approach for our demands. In order to efficiently compute the point multiplication, we used the right-to-left binary algorithm [39], a variant of the double-and-add algorithm. Our implementation on average requires 360 ms for one point multiplication.

4.3 NFC in Active Reader Mode: nPA Communication

For verifying the identity of the customer via the eID service, the NFC smartphone was set up to communicate with the nPA in active reader mode (cf. Sect. 2.2). We implemented the complete PACE protocol with Elliptic Curve Diffie-Hellman (ECDH) (cf. Sect. 2.1), employing our ECC implementation (cf. Sect. 4.2) for the ECC computations. Furthermore, we realized the required encryption and decryption with 3DES in CBC mode.

The correctness of the implementation has been successfully verified by means of test vectors [6] and practical tests with different nPAs. The achieved performance, in terms of the average runtime for 1000 PACE protocol runs is given in Table. 1. A complete PACE protocol run requires 3.5 s on average.

4.4 NFC in Card Emulation Mode: A Virtual Mifare DESFire Card

Our realization of a virtualized Mifare DESFire MF3ICD40 card employs the card emulation mode of the NFC smartphone (cf. Sect. 2.2).

We implemented the *Authenticate* procedure of the DESFire card for the initialization of the communication, the authentication, and the establishment of a secure channel between reader and smartphone. All further communication is then encrypted with 3DES in CBC mode, as done by genuine DESFire cards.

Table 1. Average runtime of our PACE implementation over 1000 protocol runs

Time	Average runtime (ms)
Up to general authentication	262
Encrypt nonce	105
Map nonce	1695
Perform key agreement	1291
Mutual authentication	148
Total	*3501*

After an authentication with key 0, the *ReadData* command is used for reading out the encrypted *User Rights Credential*. The correctness of the implementation has been validated with an ACG HF Multi ISO RFID Reader [3] that natively supports Mifare DESFire. According to our measurements, the complete process of communicating with the door lock and verifying the granted user rights takes less than one second.

4.5 Implementation Obstacles

For accessing an SE of the BlackBerry Bold 9900 smartphone, RIM can grant special programmer rights by means of two signing keys that have to be integrated into the development environment. One of these keys is the *RESE* key, required for accessing the eSE. The second key, termed *NFCR* key, is required for accessing the SE in the SIM card. Unfortunately, all our requests to RIM to get the permission for the programming part of any SE have been denied.[3] For accessing the NFC interface, we thus had to bypass the integrated SE and in consequence all computations were performed on the main processor of the smartphone — a fact which constitutes a severe security risk (cf. Sect. 5.3). Furthermore, our requests to program the main processor efficiently in C or assembly were fruitless (although technically possible), thus all programs (including the computations on elliptic curves) had to be implemented in Java.

For a practically secure realization, the required access rights to the SE are not given, the emulation of contactless cards is restricted by the hardware, the OS, and drivers, and the possibility of efficiently implementing in C (instead of slow Java implementations) is not enabled by the manufacturers and carriers. This situation is not limited to the smartphone used by us. On the contrary, it is the only NFC phone we found on which a (Mifare DESFire) card emulation is practically feasible without modifying the OS.

5 Security Analysis

In this section, we address different security aspects of the developed concept for user rights managements presented in Sect. 3, namely, the security of using the

[3] The same applies to various other manufacturers of smartphones: No vendor was willing to give us access to a secure element.

nPA without a dedicated reader, protecting the NFC interface, and performing security-relevant computations in an unprotected environment.

5.1 Security of Using the nPA with an NFC Smartphone

The security of the nPA communication depends on the implemented PACE protocol that is analyzed in [9,30] and on the EAC protocol for which security analyses can be found in [9,42]. However, a security vulnerability related to using the nPA in our implementation remains: As demonstrated in [8], for a secure implementation of the PACE protocol, the PIN has to be entered using the keypad of a dedicated passport reader [15]. In our implementation, the customer provides the PIN by typing it into the user interface of an application running on the OS of the smartphone. The OS is prone to malware, e.g., a Trojan horse could let an attacker access and modify user inputs and other data. An attacker could obtain the secret PIN and, if the smartphone is situated close to the nPA[4] and NFC is enabled, the customer could be impersonated by the attacker. Despite the impersonation attack, the attacker has no access to the private key of the customer, if it was stored in the smartphone's SE. This private key is required for sending a service request to the service provider, thus the attacker cannot fraudulently obtain credentials. Nonetheless, typing a secret into the the OS poses a severe security risk for nPA applications in general, and should never be implemented in a practical application. Securely using eID cards like the nPA with an NFC smartphone thus needs to be further researched — the currently available hardware and software cannot ensure a secure usage.

5.2 NFC Communication

Eavesdropping and manipulation of any wireless communication interface is relatively easy. This of course also applies to ISO 14443 and thus, the NFC interface [19,24,36]. In [34], the NFC attack surface is summarized. References [20,25] describe several well-known security threats of NFC. The NFC channel is secured by reliable cryptographic mechanisms in all parts of our concept, namely, those of the nPA and of Mifare DESFire cards that prevent these attacks.

A Man-In-The-Middle or relay attack is feasible for contactless smartcards and NFC [23,31], even with cryptographically secured messaging. However, the achievable ranges are well below 30 cm. An explanation of a practical relay attack on contactless transactions using NFC mobile phones is given in [21]. In the context of real-world-applications, e.g., our car sharing scenario, it is unlikely that an attacker gets into the required direct vicinity of the victim for unauthorizedly using the rental car. However, our implementation makes such an attack highly unlikely, since a user interaction via the application on the smartphone is required for initiating the communication to the car.

[4] It is conceivable that a smartphone resides close to the nPA in a wallet of the owner.

5.3 Missing Access to Secure Elements

The lack of an SE in our implementation results in serious consequences for the security of the user rights management. In principle, all relevant calculations and confidential data can be monitored by an attacker infiltrating the smartphone with malware. In our proof-of-concept implementation, an attacker could thus obtain the private key of the customer's smartphone and the credential, and use a booked car instead of the legitimate user. Nonetheless the attacker can not get newly requested credentials by herself; the attacker can send service requests on the behalf of the customer, but is not able to correctly authenticate with nPA and PIN, if the PIN entering is secured.

The worst case for user right management system presented in this paper would occur, if the smartphone SE access is denied by the manufacturer, the smartphone is infiltrated by attackers malware, so that all credentials, the secure customer key sk_C and the nPA PIN can be spied out and the nPA is in NFC communication distance. The attacker can use old credentials, send new service requests and can impersonate the customer using the nPA and the nPA PIN.

6 Conclusion and Discussion

The two main goals of this work were to develop a concept for the secure management of rights and the design and realization of a mobile car sharing application on a smartphone. We presented a concept using a state-of-the-art NFC smartphone and the nPA that is also suitable for scenarios in which no permanent Internet connection is available. We combined and extended standard protocols and implemented the scheme in practice.

In our implementation, the smartphone in one moment acts as an RFID reader to interact with other NFC-enabled objects, such as electronic ID cards and NFC door locks, and in the next moment emulates a virtualized contactless card, in our case the Mifare DESFire card. Various other existing smartphone applications and future visions in the context of NFC can benefit from the presented work, e.g., smart metering for electronic vehicles, safe deposits of virtual warehouses, managing fleets of vehicles, or booking electronic keys for hotel rooms.

We showed that the developed concept and the practical performance of the implementation is suitable for real-world applications and discussed attacks and other security issues. We further pointed out the difficulties we experienced with commercial smartphones, SEs, and other implementation obstacles that hinder the development of secure NFC implementations. These are probably important reasons why many years after its invention, NFC is still not used to its full potential.

6.1 Future Work

The concept presented in this paper offers a lot of further extensions and possibilities for future solutions. As treated in Sect. 5 some difficulties in the implementation of parts of the concept, because of missing access permissions, etc., lead to attack vectors that have to be prevented.

The missing access to the smartphone eSEs for developers is a highly relevant problem in practice that can be result in security gaps, as discussed in Sect. 5.3. So, it is highly recommended that manufacturers of smartphones, restricting the access to secure elements or the NFC interface, provide a way to flexibly use their platforms to enable the development of secure NFC solutions in the scientific community.

Another security issue of the presented concept for right management is caused by the unsecured PIN entered into the customer's smartphone. So, another very important point on future work is the development of a system for secure smartphone inputs. This can be realized for example using a Trusted Platform Module (TPM), e.g., developed by the Trusted Computing Group [40], that provides special security features, e.g., a secure boot process. This way the PIN entering can be possibly secured from malware and OS manipulation on the smartphone, while Internet access is provided nonetheless.

References

1. Emulating a PKI smart card with CyanogenMod 9.1. http://nelenkov.blogspot.it/2012/10/emulating-pki-smart-card-with-cm91.html
2. Eastlake III, D., Hansen, T.: RFC 4634 - US Secure Hash Algorithms (SHA and HMAC-SHA). Motorola Labs and AT&T Labs, July 2006
3. ACG id. HF Multi ISO RFID Reader User Manual (2006)
4. BlackBerry. BlackBerry Bold 9900/9930 Smartphones - Safety and Product Information
5. BlackBerry Support Community. Java Development - NFC Primer for Developers. http://supportforums.blackberry.com/t5/Java-Development/NFC-Primer-for-Developers/ta-p/1334857
6. Bundesamt für Sicherheit in der Informationstechnik. Worked Example for Extended Access Control (EAC) (PACE, Chip Authentication and Terminal Authentication) (2010)
7. Carlisle Adams, S.L.: Understanding PKI: Concepts, Standards, and Deployment Considerations. Pearson Education Inc., Boston (2003)
8. Chaos Computer Club. Praktische Demonstration erheblicher Sicherheitsprobleme bei Schweizer SuisseID und deutschem elektronischen Personalausweis (2010)
9. Christina Brzuska, M.F., Dagdelen, Ö.: TLS, PACE, and EAC: A Cryptographic View at Modern Key Exchange Protocols
10. CyanogenMod. http://www.cyanogenmod.org/
11. Dierks, T., Rescorla, E.: RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2. RTFM, IETF, August 2008
12. Dworkin, M.: NIST Special Publication 800–38B (Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication). NIST (2005)
13. Federal Information Processing Standards. Federal Information Processing Standards Publication 180–2 - Secure Hash Standard, February 2004
14. Federal Ministry of the Interior, German Federal Republic. White Paper - Neuer Personalausweis - eID-Server und eID-Service (2011)

15. Federal Office for Information Security, German Federal Republic. Technical Guideline TR-03119 (Requirements for Smart Card Readers Supporting eID and eSign Based on Extended Access Control) (2011)
16. Federal Office for Information Security, German Federal Republic. Technical Guideline TR-03127 (Architecture electronic Identity Card and electronic Resident Permitl) (2012)
17. Federal Office for Information Security, German Federal Republic. TR-03110-2 (Advanced Security Mechanisms for Machine Readable Travel Documents – Part 2 – Extended Access Control Version 2 (EACv2), Password Authenticated Connection Establishment (PACE), and Restricted Identification (RI)) (2012)
18. Federal Office for Information Security, German Federal Republic. TR-03110-3 (Advanced Security Mechanisms for Machine Readable Travel Documents – Part 3 – Common Specifications) (2012)
19. Finke, T., Kelter, H.: Radio Frequency Identification - Abhörmöglichkeiten der Kommunikation zwischen Lesegerät und Transponder am Beispiel eines ISO14443-Systems. BSI - Bundesamt für Sicherheit in der Informationstechnik
20. Finkenzeller, K.: RFID-Handbuch, 3rd edn. Hanser Fachbuchverlag, München (2002)
21. Francis, L., Hancke, G., Mayes, K., Markantonakis, K.: Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones, IACR Cryptology ePrint Archive (2012)
22. Gartner: Gartner Says Asia/Pacific Led Worldwide Mobile Phone Sales to Growth in First Quarter of 2013. http://www.gartner.com/newsroom/id/2482816 (2013)
23. Hancke, G.: A practical relay attack on ISO 14443 proximity cards. http://www.cl.cam.ac.uk/~gh275/relay.pdf (2005)
24. Hancke, G.: Eavesdropping attacks on high-frequency RFID tokens. In: RFIDSec (2008)
25. Haselsteiner, E., Breitfuss, K.: Security in Near Field Communication (NFC) - Strengths and Weaknesses. In: RFIDSec, Graz, Austria (2006)
26. INSIDE Secure. SecureRead. http://www.insidesecure.com/eng/Products/NFC-Products/SecuRead
27. International Civil Aviation Organization. JTC1 SC17 WG3/TF5 (Supplemental Access Control for Machine Readable Travel Documents) (2010)
28. International Organization for Standardization (ISO). ISO/IEC 14443 Parts 1–4 (2001). http://www.iso.ch
29. International Organization for Standardization/International Electrotechnical Commission. ISO/IEC 18092 (Near Field, Communication (NFCIP-1)) (2004)
30. Jens Bender, D.K., Fischlin, M.: Security Analysis of the PACE Key-Agreement Protocol (2009)
31. Kasper, T., Carluccio, D., Paar, C.: An Embedded System for Practical Security Analysis of Contactless Smartcards. In: Sauveron, D., Markantonakis, K., Bilas, A., Quisquater, J.-J. (eds.) WISTP 2007. LNCS, vol. 4462, pp. 150–160. Springer, Heidelberg (2007)
32. Kasper, T., von Maurich, I., Oswald, D., Paar, C.: Chameleon: A Versatile Emulator for Contactless Smartcards. In: Rhee, K.-H., Nyang, D. (eds.) ICISC 2010. LNCS, vol. 6829, pp. 189–206. Springer, Heidelberg (2011). http://sourceforge.net/projects/chameleon14443
33. Lochter, M., Merkle, J.: RFC 5639 - Elliptic Curve Cryptography (ECC) Brainpool Standard - Curves and Curve Generation. Federal Office for Information Security of the German Federal Republic, secunet Security Networks, IETF, March 2010

34. Miller, C.: Exploring the NFC Attack Surface. Black Hat (2012)
35. NXP. Mifare DESFire Short Form Specification MF3 IC D40. http://www.nxp.com/acrobat_download/other/identification/SFS075530.pdf (2004)
36. NXP. AN200701: ISO/IEC 14443 Eavesdropping and Activation Distance. Technical report (2007)
37. Research In Motion (RIM). BlackBerry JDE 7.0.0 API Reference. http://www.blackberry.com/developers/docs/7.0.0api
38. Research In Motion (RIM) - BlackBerry Support Community. Java Development - NFC - Virtual Target Emulation. http://supportforums.blackberry.com/t5/Java-Development/NFC-Virtual-Target-Emulation/ta-p/1509687
39. Rivain, M.: Fast and regular algorithms for scalar multiplication over elliptic curves. IACR Cryptology ePrint Archive, 338 (2011)
40. Trusted Computing Group. TPM MOBILE with Trusted Execution Environment for Comprehensive Mobile Device, Security (2012)
41. Winter, J., Wiegele, P., Pirker, M., Toegl, R.: A flexible software development and emulation framework for ARM TrustZone
42. Özgür Dagdelen, M.F.: Security Analysis of the Extended Access Control Protocol for Machine Readable Travel Documents (2010)

Protocols and Attacks

Desynchronization and Traceability Attacks on RIPTA-DA Protocol

Nasour Bagheri[1](✉), Praveen Gauravaram[2], Masoumeh Safkhani[3], and Somitra Kumar Sanadhya[4]

[1] Electrical Engineering Department, Shahid Rajaee Teacher Training University, Tehran, Iran
NBagheri@srttu.edu

[2] Innovation Labs Hyderabad, Tata Consultancy Services Limited, Hyderabad, India
P.Gauravaram@tcs.com

[3] Electrical Engineering Department, Iran University of Science and Technology, Tehran, Iran
M_Safkhani@iust.ac.ir

[4] Indraprastha Institute of Information Technology, Delhi, India
Somitra@iiitd.ac.in

Abstract. Recently Gao et al. proposed a lightweight RFID mutual authentication protocol [3] to resist against intermittent position trace attacks and desynchronization attacks and called it RIPTA-DA. They also verified their protocol's security by data reduction method with the learning parity with noise (LPN) and also formally verified the functionality of the proposed scheme by Colored Petri Nets. In this paper, we investigate RIPTA-DA's security. We present an efficient secret disclosure attack against the protocol which can be used to mount both de-synchronization and traceability attacks against the protocol. Thus our attacks show that RIPTA-DA protocol is not a RIPTA-DA.

Keywords: RFID Security · Disclosure attack · Intermittence position trace attack · Desynchronization attack

1 Introduction

An RFID system typically includes a reader and a number of tags, which may range from the expensive and battery-powered tags (active tags) with Wi-Fi capabilities to the low-cost tags that are quite constrained in resources and have no internal power (passive tags). ISO-18000-6c [5] is one of the important standards for RFID passive tags that proposes a mutual authentication protocol to communicate with passive RFID tags. However, the original protocol in the standard is known to be insecure [8]. To improve the security of this standard several protocols have been proposed in compliance to this standard, e.g., SASI [4], Gossamer [6], RAPP [13] and EMAP [7]. In this direction, Gao et al. [3] have recently discussed the security concerns of RFID systems and proposed a lightweight protocol to protect ISO-18000-6c against desynchronization attacks and intermittent position trace attacks, for which a man in the middle adversary aims to trace the tag holder, and named it "resisting the intermittent position trace attacks and desynchronization attacks (RIPTA-DA)". RIPTA-DA does not use any classical cryptographic primitive such as a block cipher, a stream cipher or a hash function to meet the passive RFID tags restrictions. The only nonlinear function which is used to provide the desired confusion and protect secret parameters of the tag is a function called "square random number function". In this function the main source of nonlinearity is number squaring calculation. More precisely, given two n-bit values, the "square random number function" computes some linear calculation combined with squaring a number and returns an n-bit result. The protocol is a lightweight protocol as it is compliant to passive tags. Hence it could be a promising solution to strengthen the security of the tags confirming ISO-18000-6c if it could provide an accepted level of security against intermittent adversary which was the main goal of the designers of the protocol.

In this paper we show that RIPTA-DA protocol is not secure against an active adversary which is able to impersonate the reader and sends several consecutive queries to a passive RFID tag. We present an efficient secret disclosure attack which, given 512 consecutive queries to the tag and its responses, retrieves more than n bits out of a $3n$-bit secret key with the success probability of almost 1. In addition, given the recovered secret, we present an approach to trace the tag for which the adversary's advantage is 0.738 for each query to the tag. Moreover we present a desynchronization attack, which after two queries to the tag, desynchronizes the tag and the reader with the probability of 1, thus they do not authenticate each other anymore. This attack contradicts the claims on the security of the RIPTA-DA protocol against desynchronization attack.

The rest of the paper is organised as follows: In Sect. 2 we present a high-level discussion on the attacks considered in this paper. In Sect. 3 we review RIPTA-DA protocol. In Sect. 4 we present secret disclosure attack on RIPTA-DA. In Sects. 5 and 6, we show how to use the revealed secrets to mount traceability attack and desynchronization attack against the protocol. Section 7 concludes the paper.

2 General Overview of Attacks on RFID Tags

Previous studies [1,9–12] discussed several threats to RFID applications, e.g., eavesdropping, replay attack, cloning, tag impersonation, secret disclosure attack, tag tracing, data forging, denial of service and counting attack. In this section we review the attacks which we mount against RIPTA-DA protocol.

Secret disclosure attack: In an RFID system, the tags and the readers have secret parameters that the adversary should not be able to discover with a complexity less that searching for them with brute force. However, if the messages transferred in a protocol are not designed properly then the adversary may reveal these secrets parameters partially or completely.

Traceability attack: In an RFID system tags and readers interact in protocol sessions. Since the communications take place over a wireless channel, it is assumed that an adversary can control any communications among all the participants and can interact passively or actively with them. In this scenario, if an adversary finds any meaningful relation between the messages transferred in different sessions of the protocol with non-negligible success probability then it succeeds in tracing the tag, thus compromising the tag holder's privacy. For example, if a specific tag always returns its static ID-value as a part of its response to the reader's query then this value can be used as a measure to trace the tag and apply a traceability attack against the protocol.

Desynchronization attack: Desynchronization attack in an RFID protocol is a type of denial of service attack. Desynchronization attack occurs if a legitimate tag and a legitimate reader do not authenticate each other because of not receiving the expected response while communicating. In RFID protocols, to provide anonymity, it is common to update some of the shared parameters among the protocol parties. In this case, a promising approach to desynchronize the tag and the reader is to force the tag and the reader to update their common values to different values. If the adversary can succeed in forcing the tag and the reader to do so, they will not authenticate each other in further transactions.

3 RIPTA-DA Protocol

RIPTA-DA protocol was proposed by Gao et al. [3] to fix the security concerns of ISO-18000-6c compliant protocols. Assume that $(\mathcal{X})_{i\sim j}$ indicates the fraction of a string $\mathcal{X}$ from the ith bit to the jth bit, $A \in \{0,1\}^n$ and $B \in \{0,1\}^n$. We explain the protocol with a discussion on the functionality of its building block, called 'square random function' and denoted by $S(A \oplus B)$, as below:

$$\begin{aligned}
x &= A \oplus B;\\
y &= x^2;\\
z &= (y)_{((B)_{k-1\sim 0})\sim(((B)_{k-1\sim 0})-(n-1))};\\
S(A \oplus B) &= z;
\end{aligned}$$

where k is a fixed value and the factory determines this k such that $k \leq log_2(n)$, $x \in \{0,1\}^n$, $z \in \{0,1\}^n$ and $y \in \{0,1\}^{2n}$. In this function, bit extraction is performed modulus the bit length of string $\mathcal{X}$, i.e., the bit extraction is done in a circular modulus; if the extraction reaches the end of the string with less than n bits, several bits will be taken from the beginning of the source string to make the length of the extracted string to be n bits. The right-most bit of $\mathcal{X}$ is the least significant bit and indexed by '0'.

For example, assume that $n = 16$, $A = 1011\ 0100\ 1110\ 0101$, $B = 0100\ 0111\ 0101\ 1101$ and $k = 3$, then:

$$\begin{aligned} x = A \oplus B &= 1111\ 0011\ 1011\ 1000; \\ y = x^2 &= 1110\ 1000\ 0000\ 0110\ 1101\ 0100\ 0100\ 0000; \\ (B)_{(3-1)\sim 0} = (B)_{(2)\sim 0} &= 101; \\ z = (y)_{5\sim(-10)} &= 0000\ 00\ 1110\ 1000\ 00; \\ S(A \oplus B) &= 0000\ 0011\ 1010\ 0000. \end{aligned}$$

In RIPTA-DA protocol, the tag and the reader share three n-bit secret keys denoted by key_iH, key_iM and key_iL, where i is the session index. In this protocol, to avoid desynchronization attacks, both the tag and the reader keep two records of the secret parameters denoted by key_i1 and key_i2 respectively where key_i is a key group that includes $\{key_iH, key_iM, key_iL\}$ and is updated after each successful run of the protocol. In addition, each tag has a single bit $flag$ and a judgment function that could be used to determine which of key_i1 or key_i2 of the secret key group is the last successful authentication secret key group. More precisely, $flag = 1$ implies that key_i1 group keeps the previous success authentication secret key group and $flag = 0$ implies that key_i2 group keeps the previous successful secret key group. The purpose of this bit is to ensure that the correct secret key group of a mutual authentication will always be kept in any situation between the reader and the tag. This is aimed at avoiding desynchronization attack. The $flag$ bit is updated by Update($flag$) function. In addition, on each round of the protocol a random number v is contributed to the calculation of the transferred messages by the tag. The designers of the protocol claim that this random value plays the role of the noise in the LPN based protocols [3, p. 1950], aiming to employ the LPN to enhance the protocol security functionality.

RIPTA-DA protocol which is shown in Fig. 1 runs as follows:

1. The reader sends $Query$ command with a random number N, generated by database, to the tag T.
2. Upon receiving the message, T does as follows:
   - generates a random number v,
   - computes $R = S(key_iH \oplus v)$, $\alpha = key_iM \oplus v, \beta = key_iL \oplus R$, and $\mu = H(N \oplus key_iH \oplus v)$, where H is a hash function[1] The effective key group is key_i1 group if $flag = 1$, otherwise it is key_i2.
   - and sends the tuple $\{\alpha, \beta, \mu\}$ to the reader.
3. The reader receives the message sent by the tag and transmits it to the database.

[1] We note that [3] does not clearly state that $H(.)$ is a hash function. However, for other protocols discussed in this paper the authors have used this notation for a hash function, e.g. [3, p. 1951]. Therefore we take it that H denotes a hash function in the calculation of μ. It must be noted that the details of $H(.)$ have no impact on the success probability of the attacks presented in this paper.

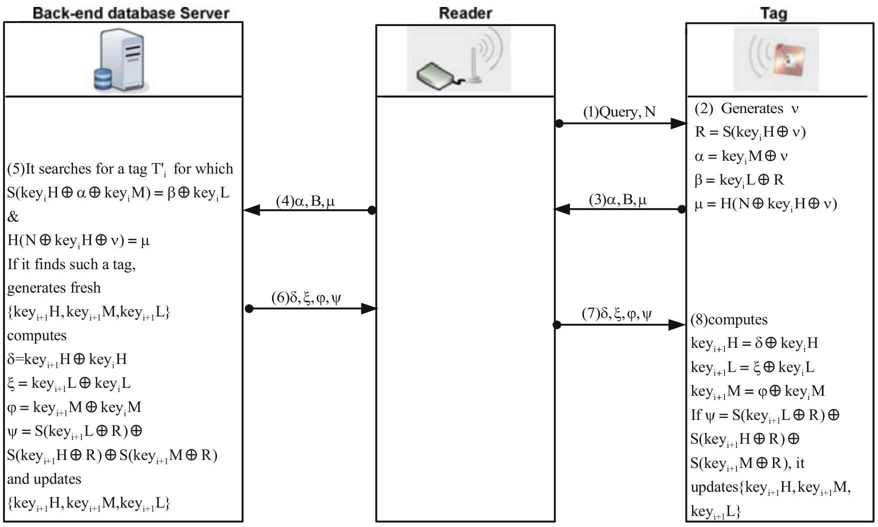

Back-end database Server
Reader
Tag
(1)Query, N
(2) Generates ν
R = S(key_iH ⊕ ν)
α = key_iM ⊕ ν
β = key_iL ⊕ R
μ = H(N ⊕ key_iH ⊕ ν)
(3)α, B, μ
(4)α, B, μ
(5)It searches for a tag T'_i for which S(key_iH ⊕ α ⊕ key_iM) = β ⊕ key_iL & H(N ⊕ key_iH ⊕ ν) = μ
If it finds such a tag, generates fresh {key_{i+1}H, key_{i+1}M, key_{i+1}L} computes
δ=key_{i+1}H ⊕ key_iH
ξ = key_{i+1}L ⊕ key_iL
φ = key_{i+1}M ⊕ key_iM
ψ = S(key_{i+1}L ⊕ R) ⊕ S(key_{i+1}H ⊕ R) ⊕ S(key_{i+1}M ⊕ R)
and updates {key_{i+1}H, key_{i+1}M, key_{i+1}L}
(6)δ, ξ, φ, ψ
(7)δ, ξ, φ, ψ
(8)computes
key_{i+1}H = δ ⊕ key_iH
key_{i+1}L = ξ ⊕ key_iL
key_{i+1}M = φ ⊕ key_iM
If ψ = S(key_{i+1}L ⊕ R) ⊕ S(key_{i+1}H ⊕ R) ⊕ S(key_{i+1}M ⊕ R), it updates{key_{i+1}H, key_{i+1}M, key_{i+1}L}

Fig. 1. RIPTA-DA mutual authentication protocol.

4. Upon receiving the message, database does as follows:
   – It searches for a tag T' for which $S(key_iH \oplus \alpha \oplus key_iM) = \beta \oplus key_iL$ and $H(N \oplus key_iH \oplus v) = \mu$.
   – If it finds such a tag, it authenticates the tag, generates fresh $\{key_{i+1}H, key_{i+1}M, key_{i+1}L\}$, chosen uniformly at random from the key space, computes $\delta = key_{i+1}H \oplus key_iH$, $\xi = key_{i+1}L \oplus key_iL$, $\varphi = key_{i+1}M \oplus key_iM$ and $\Psi = S(key_{i+1}L \oplus R) \oplus S(key_{i+1}H \oplus R) \oplus S(key_{i+1}M \oplus R)$ and sends the tuple $(\delta, \varphi, \xi, \Psi)$ to the reader,
   – updates $\{key_{i+1}H, key_{i+1}M, key_{i+1}L\}$.
5. The reader transfers the received $(\delta, \varphi, \xi, \Psi)$ to the tag.
6. Upon receiving the message, T extracts $\{key_{i+1}H, key_{i+1}M, key_{i+1}L\}$ by using the variables δ, φ, ξ and verifies the correctness of Ψ using the extracted values. If Ψ is valid, the tag updates the non effective secret group by the extracted data, keeps the effective key_i group and adopts the $flag$ bit to indicate the secret group which has been involved in this run of protocol.

The designers of RIPTA-DA claim that their protocol provides optimal security against desynchronization attack and traceability attack. However, in this paper we show that the protocol is weak by presenting a secret disclosure attack which can be employed to desynchronize the tag and the reader and also trace the tag's holder.

4 Secret Disclosure Attack on RIPTA-DA

The only source of nonlinearity in RIPTA-DA protocol is *square random number function* which is mainly squaring an n-bit value and dropping n bits of

Table 1. $(\mathcal{X}^2)_{2\sim0}$ given $(\mathcal{X})_{2\sim0}$. Note that $(\mathcal{X})_{n-1\sim3}$ has no effect on the result of $(\mathcal{X}^2)_{2\sim0}$

$(\mathcal{X})_2$	$(\mathcal{X})_1$	$(\mathcal{X})_0$	$(\mathcal{X}^2)_2$	$(\mathcal{X}^2)_1$	$(\mathcal{X}^2)_0$
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	1	0	0
0	1	1	0	0	1
1	0	0	0	0	0
1	0	1	0	0	1
1	1	0	1	0	0
1	1	1	0	0	1

the result. However, by a close look at $\mathcal{X}^2 = \mathcal{X} \times \mathcal{X}$ we see some interesting properties that can be employed through our analysis. Assuming that $\mathcal{X} = (\mathcal{X})_{n-1}\|\ldots\|(\mathcal{X})_1\|(\mathcal{X})_0$, where $(\mathcal{X})_i$ denotes the ith bit of $\mathcal{X}$, the following properties hold between the bits of $\mathcal{X}$ and the bits of $\mathcal{X}^2$, also see Table 1:

$$(\mathcal{X}^2)_0 = (\mathcal{X})_0; \tag{1}$$

$$(\mathcal{X}^2)_1 = 0; \tag{2}$$

$$(\mathcal{X}^2)_2 = (\mathcal{X})_1.\overline{(\mathcal{X})_0}. \tag{3}$$

On each query to the tag T by a reader, which can potentially be an adversary as well, the tag returns the following values:

$$\alpha = key_iM \oplus v; \tag{4}$$

$$\beta = key_iL \oplus R. \tag{5}$$

where $R = S(key_iH \oplus v)$. In addition, as long as the tag has not been involved in a successful run of the protocol, the effective secret key remains fixed. We assume the adversary initiates t consecutive sessions. In the jth session, the adversary sends N^j to the tag and the tag answers the query by $\{\alpha^j, \beta^j, \mu^j\}$, where:

$$\alpha^j = key_iM \oplus v^j;$$
$$\beta^j = key_iL \oplus R^j;$$
$$R^j = S(key_iH \oplus v^j).$$

for $1 \le j \le t$. On the other hand, for $1 \le m \le n$ as a bit position, $1 \le j \le t$ and $1 \le f \le t$, we have:

$$(\alpha^j \oplus \alpha^f)_m = (v^j \oplus v^f)_m = (v^j)_m \oplus (v^f)_m$$

Since the attacker knows α^j and α^f, he can easily determine $(v^j)_m \overset{?}{=} (v^f)_m$. Given this information, it is possible to group $v^1, \ldots, v^t$ into two groups, denoted by $G1$ and $G2$, respectively, where any entry in a group holds the same value in its mth bit of v. For example, given $(v^1)_m \oplus (v^j)_m$, for $1 \le j \le t$, it is possible to assign to $G1$ any entry that contains $(v)_m$ equal to $(v^1)_m$, i.e., $(v^1)_m \oplus (v^j)_m = 0$, and assign to $G2$ any entry that contains $(v)_m$ not equal to $(v^1)_m$, i.e., $(v^1)_m \oplus (v^j)_m = 1$. Similar approach can be used to group $v^1, \ldots, v^t$ into 2^k groups, denoted by $G1, \ldots, G2^k$, respectively, where any entry in a group holds the same value in its k least significant bits of v, i.e., $(v)_{k-1\sim 0}$. Next, we look for a group for which the k least significant bits of v are equal to $n-1$. For such a value of v, $R = S(key_i H \oplus v)$ is calculated as follows:

$$\begin{aligned} x &= key_i H \oplus v; \\ y &= x^2; \\ R &= (y)_{(n-1)\sim 0}. \end{aligned} \tag{6}$$

Hence $(R)_{2\sim 0} = ((key_i H \oplus v)^2)_{2\sim 0}$, where based on Eq. (2) we have $(R)_1 = 0$ and therefore $(\beta^j)_1 = (key_i L \oplus R)_1 = (key_i L)_1$. Based on this observation, if for a group Gi, $(v)_{k-1\sim 0} = n-1$, then for all elements of that group $(\beta)_1$ should remain constant. Given such a group we have revealed $(key_i L)_1$ and $(v)_{k-1\sim 0} = n-1$. By using the second value ($(v)_{k-1\sim 0} = n-1$) which is revealed we can determine k bits of $key_i M$ because $(\alpha^j)_{k-1\sim 0} = ((key_i M) \oplus (n-1))_{k-1\sim 0}$. Given $(key_i M)_{k-1\sim 0}$ it is possible to determine $(v)_{k-1\sim 0}$ of each group. It must be noted the extracted bits of v contradict the claimed reduction for the security of the protocol to LPN problem because in the LPN problem the adversary's advantage to receive any information related to the noise parameter v should be negligible otherwise Gaussian elimination to obtain the secret parameter would be possible. So far, for each group Gi, for $1 \le i \le 2^k$, we know where the location of $((key_i H \oplus v)^2)_1 = 0$ would be to determine another bit of $key_i L$. Following this approach, we can determine 2^k bits of the secret parameter $key_i L$ and also $(key_i M)_{k-1\sim 0}$ (if $2^k = n$ then we can retrieves all bits of $key_i L$). Given $key_i L$, the adversary can determine R as $R = \beta \oplus key_i L$. On the other hand, based on Eq. 1 $(x^2)_0 = x_0$ which combined with the extracted $(v)_0$ reveals $(key_i H)_0$. In addition, given $(key_i H)_0$, $(v)_1$ and Eq. (3), the adversary retrieves $(key_i H)_1$. (This attack is a partial key recovery attack for $key_i H$. Although it is possible to extend this attack to recover up to k bits of $key_i H$ given $(v)_{0\sim k}$ and R for the eavesdropped messages in the previous sessions and some simple calculations, it is not needed for our attacks in the rest of the paper.)

The adversary succeeds in her attack if she selects a correct group in the first step of the attack, as a group for which $(v)_{k-1\sim 0} = n-1$. On the other hand, if for a group $(v)_{k-1\sim 0} = n-1$ then for all elements of that group $(\beta)_1$ would be constant and if $(v)_{k-1\sim 0} \neq n-1$ all elements of that group holds the same value of $(\beta)_1$ only with the probability of $2^{-|G|}$, where $|G|$ denotes the group's cardinality which is approximately $\frac{t}{2^k}$. Hence, excluding the correct group, the adversary is expected to receive $(|\#G|-1) \times 2^{-|G|}$ groups that satisfy the given condition on $(\beta)_1$, where $|\#G|$ denotes the total number of groups ,i.e, 2^k. We call such a group a quasi-correct-group. In addition, after this step the adversary knows the expected value of $(v)_{k-1\sim 0}$ for each group. This value can be used to determine the location of $(key_i L)_1$ in each group which in turn can be used to filter wrong guesses. The adversary fails in her attack if all the given conditions are satisfied for a wrongly selected group. For a quasi-correct-group all bits in the expected location for $(key_iL)_1$ on each group holds with the probability of $2^{-|G|}$. We have 2^k groups and $(|\#G|-1) \times 2^{-|G|}$ quasi-correct-groups. So a quasi-correct-group passes all conditions with the following probability:

$$((|\#G|-1) \times 2^{-|G|}) \times \left(2^{-|G|}\right)^{\#|G|} = (2^k-1) \times 2^{-\frac{t}{2^k}} \times \left(2^{-\frac{t}{2^k}}\right)^{2^k}.$$

As a numerical example, for $l = 128$, $k = 8$ and $t = 512$, the group's cardinality is expected to be $\frac{512}{128} = 4$ and a quasi-correct-group passes the given conditions with the probability of $(2^8-1) \times 2^{-4} \times \left(2^{-4}\right)^{128} \cong 2^{-508}$. Hence the adversary's advantage in the given attack for $t \geq 512$ is almost 1.

5 Traceability Attack

Given the target tag T and its secret $(key_iH)_{1\sim0}$, $(key_iM)_{k-1\sim0}$ and key_iL, to determine whether a randomly selected tag T' is T, the adversary initiates a session by sending a random number N and receives tuples $\{\alpha, \beta, \mu\}$ and does the following calculations:

$$R' = key_iL \oplus \beta;$$
$$(v')_{k-1\sim0} = (key_iM \oplus \alpha)_{k-1\sim0}.$$

Given $\beta = key_iL \oplus R'$ and key_iL we can compute R'. Given α and $(key_iM)_{k-1\sim0}$ we can determine $(v')_{k-1\sim0}$. Given $(v')_{k-1\sim0}$ we know the value of $((key_i'H \oplus v')^2)_{2\sim0}$, which combined with $(v')_{k-1\sim0}$ allows us to determine $(key_i'H)_{1\sim0}$. Remember that $((key_i'H \oplus v')^2)_0 = (key_i'H \oplus v')_0$ and $((key_i'H \oplus v')^2)_2 = (key_i'H \oplus v')_1.\overline{(key_i'H \oplus v')_0}$. Therefore, if $(v')_{k-1\sim0} \geq 2$ then the adversary can determine $((key_i'H \oplus v')^2)_{2\sim0}$ and $(key_i'H)_{1\sim0}$. Then the adversary outputs '1' if $(key_iH)_{1\sim0} = (key_i'H)_{1\sim0}$, otherwise outputs '0'. The adversary's advantage Adv_A to make the correct decision in this attack is defined as follows:

$$Adv_A = \left| Pr[A^{T=T'} \Rightarrow 1] - Pr[A^{T\neq T'} \Rightarrow 1] \right|.$$

To determine Adv_A we need $Pr((v')_{k-1\sim0} \geq 2)$ which is $1 - \frac{2}{2^k}$. If $T = T'$ and $(v')_{k-1\sim0} \geq 2$ then with the probability of 1, the adversary outputs "1" and if $(v')_{k-1\sim0} < 2$ then it just return a random bit as its decision. on the other hand, if $T \neq T'$ and $(v')_{k-1\sim0} \geq 2$ then with the probability of $\frac{1}{4}$ the adversary outputs '1' and if $(v')_{k-1\sim0} < 2$ then it just return a random bit as its decision. Hence:

$$Adv_A = \left| (1 - \frac{2}{2^k}) \times 1 + (\frac{2}{2^k}) \times \frac{1}{2} - (1 - \frac{2}{2^k}) \times \frac{1}{4} - (\frac{2}{2^k}) \times \frac{1}{2} \right|.$$

For $k = 7$, $Adv_A = 0.738$ which is not negligible. It must be noted the adversary may repeat the above attack to increase its advantage. It is clear given a tag T', to verify whether it is the target tag T the adversary can use the approach presented in Sect. 4 to extract its secret parameters, compare them with those of T and output its decision. However, the complexity of this approach is much higher than the detailed attack in this section.

6 Desynchronization Attack

If an authentication protocol's secret parameters, that are used through the authentication process, are updated then both parties should keep the same value. Otherwise they won't authenticate each other in the later sessions and we say they have been desynchronized. In the desynchronization attack which is presented in this section, the adversary forces the tag and the back-end server to update their common values to different values. Gao et al. [3] claim that their protocol is secure against desynchronization attack. More precisely, the authors state that to prevent desynchronization attacks both the tag and the back-end database keep the latest successful authenticated group of secrets which can be used to resynchronize the tag and the server. However, based on the secret disclosure attack given in Sect. 4, we present an efficient attack where the adversary forces the tag to update both records of secret values such that neither of them matches the values that back-end database keeps in its records. Given the target tag T and its secret $(key_iH))_{1\sim 0}$, $(key_iM))_{k-1\sim 0}$ and key_iL, an active adversary ($\mathcal{A}$), which is present during the communication of the tag and the reader follows a two phased attack to desynchronize the tag and the reader.

Phase 1 (**updating** key_{i+1}): In this phase of attack, the adversary $\mathcal{A}$ forces the tag and the database to update their record of $key_{i+1}L$ to different values as follows:

1. The reader sends *Query* command with a random number N, generated by database, to the tag T.
2. Upon receiving the message, T does as follows:
   - generates a random number v,
   - computes $R = S(key_iH \oplus v)$, $\alpha = key_iM \oplus v, \beta = key_iL \oplus R$, and $\mu = H(N \oplus key_iH \oplus v)$, where if $flag = 1$ then key_i1 group is the effective secret key group; otherwise key_i2,
   - and sends the tuple $\{\alpha, \beta, \mu\}$ to the reader.
3. $\mathcal{A}$ eavesdrops the message and extracts R from $\beta = key_iL \oplus R$.
4. The reader receives the message sent by the tag and transmits it to the database.
5. Upon receiving the message, database does as follows:
   - It searches for a tag T' for which $S(key_iH \oplus \alpha \oplus key_iM) = \beta \oplus key_iL$ and $H(N \oplus key_iH \oplus v) = \mu$.
   - It finds T and generates fresh $\{key_{i+1}H, key_{i+1}M, key_{i+1}L\}$ uniformly at random from the key space, computes $\delta = key_{i+1}H \oplus key_iH$, $\xi = key_{i+1}L \oplus key_iL$, $\varphi = key_{i+1}M \oplus key_iM$ and $\Psi = S(key_{i+1}L \oplus R) \oplus S(key_{i+1}H \oplus R) \oplus S(key_{i+1}M \oplus R)$ and sends the tuple $(\delta, \xi, \varphi, \Psi)$ to the reader,
   - updates $\{key_{i+1}H, key_{i+1}M, key_{i+1}L\}$.
6. The reader transfers the received $(\delta, \xi, \varphi, \Psi)$ to the tag.
7. $\mathcal{A}$ blocks the message, extracts $key_{i+1}L$ from ξ, chooses a random $key'_{i+1}L \neq key_{i+1}L$ and computes $\xi' = key'_{i+1}L \oplus key_iL$ and $\Psi' = \Psi \oplus S(key_{i+1}L \oplus R) \oplus S(key'_{i+1}L \oplus R) = S(key_{i+1}L \oplus R) \oplus S(key_{i+1}H \oplus R) \oplus S(key_{i+1}M \oplus R) \oplus S(key_{i+1}L \oplus R) \oplus S(key'_{i+1}L \oplus R)$ and sends δ, ξ', φ and Ψ' to the tag.

8. Upon receiving the message, T extracts $\{key_{i+1}H, key_{i+1}M, key'_{i+1}L\}$ and verifies the correctness of the received Ψ'. If Ψ' is valid, which it is, the tag updates the non effective secret group by the extracted data, keeps the effective key_i group and adopts the $flag$ bit to indicate the secret group which has been involved in this run of the protocol.

Phase 2 (updating key_i**)**: In this phase of attack, which should be accomplished in the next consecutive session of protocol between T and the reader, the adversary $\mathcal{A}$ forces the tag and the database to update their record of key_iL to different values as follows:

1. The reader sends $Query$ command with a random number N', generated by database, to the tag T.
2. Upon receiving the message, T does as follows:
   - generates a random number v',
   - computes $R' = S(key_{i+1}H \oplus v')$, $\alpha' = key_{i+1}M \oplus v', \beta' = key'_{i+1}L \oplus R$, and $\mu' = H(N' \oplus key_iH \oplus v')$, where if for the previous session $flag = 0$ and key_i1 group was the effective secret key group here key_i2 is the effective one and vice versa,
   - and sends the tuple $\{\alpha', \beta', \mu'\}$ to the reader.
3. $\mathcal{A}$ blocks the message, extracts R' from $\beta' = key'_{i+1}L \oplus R'$, generates $\beta'' = key_{i+1}L \oplus R'$ and sends the tuple $\{\alpha', \beta'', \mu'\}$ to the reader.
4. The reader receives the tuple $\{\alpha', \beta'', \mu'\}$ and transmits it to the database.
5. Upon receiving the message, database does as follows:
   - It searches for a tag T' for which $S(key_{i+1}H \oplus \alpha' \oplus key_{i+1}M) = \beta'' \oplus key_{i+1}L$ and $H(N' \oplus key_{i+1}H \oplus v') = \mu'$.
   - It finds T and generates fresh $\{key'_iH, key'_iM, key'_iL\}$, computes $\delta' = key_{i+1}H \oplus key'_iH$, $\xi' = key_{i+1}L \oplus key'_iL$, $\varphi' = key_{i+1}M \oplus key'_iM$ and $\Psi' = S(key'_iL \oplus R) \oplus S(key'_iH \oplus R) \oplus S(key'_iM \oplus R)$ and sends the tuple $(\delta', \xi', \varphi', \Psi')$ to the reader,
   - updates $\{key'_iH, key'_iM, key'_iL\}$.
6. The reader transfers the received $(\delta', \xi', \varphi', \Psi')$ to the tag.
7. $\mathcal{A}$ blocks the message, extracts key'_iL form ξ', chooses a random key''_iL such that $(key''_iL \neq key'_iL)$ and $key''_iL \neq key_{i+1}L$ and computes $\xi'' = key'_{i+1}L \oplus key''_iL$ and $\Psi'' = \Psi' \oplus S(key_iL \oplus R) \oplus S(key''_iL \oplus R) = S(key'_iL \oplus R) \oplus S(key'_iH \oplus R) \oplus S(key'_iM \oplus R) \oplus S(key_iL \oplus R) \oplus S(key''_iL \oplus R)$ and sends δ', ξ'', φ' and Ψ'' to the tag.
8. Upon receiving the message, T extracts $\{key'_iH, key'_iM, key''_iL\}$ and verifies the correctness of the received Ψ''. If Ψ'' is valid, which it is, the tag updates the non effective secret group by the extracted data, keeps the effective key_{i+1} group and adopts the $flag$ bit to indicate the secret group which has been involved in this run of protocol.

At the end of this attack, the database keeps the following records of secret key groups:

$$key_i = \{key'_iH, key'_iM, key'_iL\};$$
$$key_{i+1} = \{key_{i+1}H, key_{i+1}M, key_{i+1}L\};$$

while the tag T keeps the following records as its secret key groups:

$$key_i = \{key'_i H, key'_i M, key''_i L\};$$
$$key_{i+1} = \{key_{i+1} H, key_{i+1} M, key'_{i+1} L\};$$

where $key'_{i+1} \neq key_{i+1}$ and $key''_i \neq key'_i$. Hence the adversary successfully forced the tag and the reader to update their records to different values and they won't authenticate each other in later sessions of protocol. The success probability of the presented attack is almost 1 while the complexity is just two sessions of protocol, given that the adversary has already extracted the related secrets.

An interesting property of the attack is that, when the tag and the database have been desynchronized, the only party which can resynchronize them again or let them to perform a particular session correctly is the adversary. To do so, it is enough to follow Phase 2 of the above attack.

7 Conclusion

In this paper we have shown some security pitfalls in the design of RIPTA-DA protocol. We present three attacks against the protocol. The main reason why we do not attempt to repair RIPTA-DA or design a new protocol is because we believe that it is simply not possible. We remark that it is worth investigating the design and performance aspects of RFID protocols by using standard ciphers such as PRESENT [2].

Acknowledgements. We would like to thank anonymous reviewers for useful comments.

References

1. Bagheri, N., Safkhani, M., Peris-Lopez, P., Tapiador, J.E.: Weaknesses in a new ultralightweight RFID authentication protocol with permutation-RAPP. Secur. Commun. Networks (2013). doi:10.1002/sec.803
2. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
3. Gao, L., Ma, M., Shu, Y., Wei, Y.: A security protocol resistant to intermittent position trace attacks and desynchronization attacks in RFID systems. Wirel. Pers. Commun. **68**(4), 1943–1959 (2013)
4. Hung-Yu, C.: SASI: a new ultralightweight RFID authentication protocol providing strong authentication and strong integrity. IEEE Trans. Dependable Secure Comput. **4**(4), 337–340 (2007)
5. Information technology Radio frequency identification for item management. Part 6: parameters for air interface communications at 860 MHz to 960 MHz. http://www.iso.org (2005)

6. Peris-Lopez, P., Hernandez-Castro, J.C., Tapiador, J.M.E., Ribagorda, A.: Advances in ultralightweight cryptography for low-cost RFID tags: gossamer protocol. In: Chung, K.-I., Sohn, K., Yung, M. (eds.) WISA 2008. LNCS, vol. 5379, pp. 56–68. Springer, Heidelberg (2008)
7. Peris-Lopez, P., Hernandez-Castro, J.C., Estevez-Tapiador, J.M., Ribagorda, A.: EMAP: an efficient mutual-authentication protocol for low-cost RFID tags. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2006 Workshops. LNCS, vol. 4277, pp. 352–361. Springer, Heidelberg (2006)
8. Peris-Lopez, P., Hernandez-Castro, J.C., Estevez-Tapiador, J.M., Ribagorda, A.: RFID specification revisited. In: The Internet of Things: From RFID to the Next-Generation Pervasive Networked Systems, pp. 311–346. Taylor & Francis, Bristol (2008)
9. Qian, Z., Chen, C., You, I., Lu, S.: ACSP: a novel security protocol against counting attack for UHF RFID systems. Comput. Math. Appl. **63**(2), 492–500 (2012)
10. Safkhani, M., Peris-Lopez, P., Bagheri, N., Naderi, M., Hernandez-Castro, J.C.: On the security of Tan et al. serverless RFID authentication and search protocols. In: Hoepman, J.-H., Verbauwhede, I. (eds.) RFIDSec 2012. LNCS, vol. 7739, pp. 1–19. Springer, Heidelberg (2013)
11. Sun, H.-M., Ting, W.-C.: A Gen2-based RFID authentication protocol for security and privacy. IEEE Trans. Mob. Comput. **8**(8), 1052–1062 (2009)
12. Tan, C.C., Sheng, B., Li, Q.: Secure and serverless RFID authentication and search protocols. IEEE Trans. Wireless Commun. **7**(4), 1400–1407 (2008)
13. Tian, Y., Chen, G., Li, J.: A new ultralightweight RFID authentication protocol with permutation. IEEE Commun. Lett. **16**(5), 702–705 (2012)

Long Distance Relay Attack

Luigi Sportiello(✉) and Andrea Ciardulli

European Commission, Joint Research Centre,
Via Enrico Fermi 2749, 21027 Ispra, VA, Italy
luigi.sportiello@jrc.ec.europa.eu,
andrea.ciardulli@ext.jrc.ec.europa.eu

Abstract. Contactless smart cards are used to securely store data and to authorize the execution of sensitive operations. Their contactless interface represents a mixed blessing, allowing fast operations but also exposing such devices to potential attacks. Relay attacks are among the most powerful attacks applicable against contactless smart cards, allowing a contactless reader to interact with a physically far away card establishing a communication channel between them. In this paper we prove that it is possible to conduct such an attack on a geographical scale, basically without any constraints on the reader and card positions and reaching a relay distance of several kilometers, probably the first example in the literature for contactless smart cards, using cheap and off-the-shelf hardware and software tools.

Keywords: Contactless smart cards · Relay attack · Mobile phones · NFC · Practical attack

1 Introduction

A contactless smart card is basically composed of a plastic support embedding a chip connected to a coil. Such a card is used through a reader that emits a RF field, which is exploited to power the chip by induction through the coil and to interact with it modulating commands and responses. We focus in particular on proximity contactless smart cards that have to be put in a range of few centimeters from a reader for a successful communication with it. The chip typically stores some data used by the reading system in the context of an application. In particular such cards are becoming more and more widespread and used, among others, as tokens for access control, in electronic identity documents, in payment cards and as electronic tickets.

A relay attack against a contactless smart card consists in deceiving a reader into believing that it is in proximity of such card when in fact it is not. The attack is depicted in Fig. 1. Two devices are needed, a Mole and a Proxy: the Mole has to be in proximity of the victim card while the Proxy is nearby the reader. The devices have to be featured by a contactless interface, respectively to interact with the card and with the reader, and a communication channel has to established between them. The messages sent by the reader are conveyed to

M. Hutter and J.-M. Schmidt (Eds.): RFIDsec 2013, LNCS 8262, pp. 69–85, 2013.
DOI: 10.1007/978-3-642-41332-2_5,

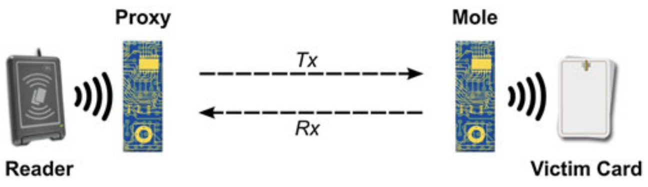


Fig. 1. Relay attack against a contactless smart card.

the card through the Proxy-Mole link and vice versa for the card messages. In such a way the reader believes to interact with the original card.

The implications of these attacks are serious, as smart cards are typically used to securely store some information and to act as token to authorize some sensitive operations. With a relay attack it is not required to clone or steal a card but only to use it remotely. Note that to guarantee the authenticity of a card some security mechanisms are usually put in place, typically running a cryptographic authentication protocol between the reader and the card that stores some secret/private keys. Relay attacks are completely able to circumvent these cryptographic mechanisms.

The idea of relaying the communication between a contactless smart card and a reader is not new in the literature [1] and some successful attacks have been published over the last years, using different equipment and different communication channels between Mole and Proxy to implement the attack. In [2,3] a specific hardware is designed to implement both the Mole and the Proxy that communicate over a dedicated RF channel, achieving a relay attack over a distance of 50 m for the former experiment. The authors of [4,5] propose a hybrid solution, with the Mole represented by a mobile phone while a specific programmable device is used as Proxy, with the two devices directly connected through a Bluetooth channel, getting a relay distance of some meters. A solution based on two mobile phones is presented in [6,7], relying respectively on a Bluetooth channel and a WiFi network for the communication, with the latter set-up able to perform card transactions with Mole and Proxy placed in two different rooms, thus relaying the communication in a building area.

Our main contribution is represented by the development of a long distance relay attack against a contactless smart card, proving the effectiveness of such an attack on a geographical scale with a relay distance of several kilometers. To the best of our knowledge, it is the first time that a relay attack is carried out against a contactless smart card over such a long distance. The attack is achieved using two mobile phones for the Mole and Proxy roles, which is not a novelty according to the references cited above, but we use the mobile phone network to relay the communication. This means that the attack, apart from the achievable long distances, is also applicable in dynamic conditions, basically without any constraints on the positions of Mole and Proxy, with the former that can potentially be even on the move. The architecture proposed for the attack allows to solve some problems induced by the use of the mobile phone network as relay channel and it is generic enough to be used even if the devices adopt other connections. We provide time measurements to highlight the delay introduced by a mobile phone network, and our architecture in particular, in the relay process. In addition the attack is particularly credible as entirely based on cheap and off-the-shelf hardware and software tools.

The paper is organized as follows. Section 2 introduces the contactless smart card communication architecture. In Sect. 3 we present the architecture developed for our attack, while Sect. 4 is for our experiments and relative results. In Sect. 5 we discuss the implications of our achievements, while Sect. 6 is for conclusions.

2 Contactless Smart Card Communication Architecture

ISO/IEC 14443 [8] is one of the most used standard in the contactless smart card field. It specifies the card-reader communication, defining the physical characteristics of the card, the features of the field used to power the card and to communicate with the reader (13.56 MHz), the anticollision procedures when multiple cards are in proximity of a reader, the format of the frames used in the communication and the half-duplex protocol used to exchange data between card and reader.

According to the standard cards can operate in a range up to approximately 10 cm from the reader, so a physical proximity is required to establish an interaction. Once a card has been put in the field of the reader, they establish a communication exchanging initialization and anticollision messages. After that, the communication is organized in a master-slave mode: the reader sends a request to the card that has to reply with a response, with this process continued until the end of the communication. The standard specifies some time constraints for this request-response interaction. In particular, before starting the interaction, card and reader agree on a maximum time allowed to return a response; to be more precise, it is the card that points out which is the maximum time it needs to prepare and return a response to a reader request. During the communication, if the reader does not receive any responses for a sent request within the specified maximum time, after a recovery attempt, it aborts the communication, as probably something has disrupted the interaction. The card points out this time through a parameter called Frame Waiting Integer (FWI), with the maximum response time that can be max $\sim$4.95 s. The FWI is typically used by the card to inform the reader that a certain amount of time will be needed to return some responses, for instance because some time consuming computations (e.g., cryptographic operations) have to be internally carried out by the chip, so as to avoid erroneous aborted communications.

Upon the communication structure provided by the ISO/IEC 14443, the card-reader interaction can be encoded through the Application Protocol Data Unit (APDU) messages defined in the ISO/IEC 7816-4 standard [9], which specifies several operations (e.g., read/write data from/to the card chip, reader/card authentication request). Such messages are packed into ISO/IEC 14443 request-response frames during the communication: the reader sends APDU commands and the chip replies with APDU responses. The messages are byte strings featured by the following format

Command:	CLA	INS	P1	P2	Lc	Data	Le
Response:	Data	SW1	SW2				

where CLA are class bytes, INS denotes the command the chip has to run, P1 and P2 are command parameters, Lc is the length of the data sent by the reader contained in the Data field, Le represents the expected length of Data in the response, and SW1-SW2 are status bytes returning information regarding the launched command (e.g., successful/unsuccessful operation). The received command is typically interpreted by the card chip, the identified operation executed and the relative response returned.

3 Relay Attack Architecture

Many modern smart phones are equipped with a Near Field Communication (NFC) interface that allow them to interact with contactless smart cards acting as a reader [10]. On the phone a reader application has to be installed, which establishes an ISO/IEC 14443 communication with the card sending APDU commands and receiving the relative responses. The NFC interface is available for several Android-based smart phones (it is also available on Nokia Lumia and BlackBerry phones but not on iPhones, even if according to some recent rumors it could be introduced soon on such devices), a platform well supported by a large community. Thanks to this wide support, the possibility to use the NFC interface of such devices, not only as card reader, but also as card emulator [11], has been recently developed and pointed out. Indeed, replacing the original phone firmware with the CyanogenMod aftermarket version [12], the NFC interface becomes available for software card emulation purposes [13]. After such a firmware upgrade, it is indeed possible to develop and install on the phone a card emulation application, which is able to read APDU commands received on the NFC interface from a contactless reader, parse and interpret such commands, perform the associated operations and reply with the appropriate responses through the NFC interface. For all the above-mentioned reasons we decided to focus on the Android platform for the experiments we present below, but we remark that the solution developed is general and applicable to all those devices able to operate in smart card emulation mode.

Our aim was to develop a relay attack proof of concept based on two NFC-enabled mobile phones connected together exploiting their Internet connection. Most of the mobile phones used nowadays are indeed connected to the Internet through a data network offered by their mobile phone network operator, which assigns an IP address to each device. For both devices a specific application has to be developed and installed: for the phone acting as Mole the application has to put the device in contactless reader mode, while for the one acting as Proxy

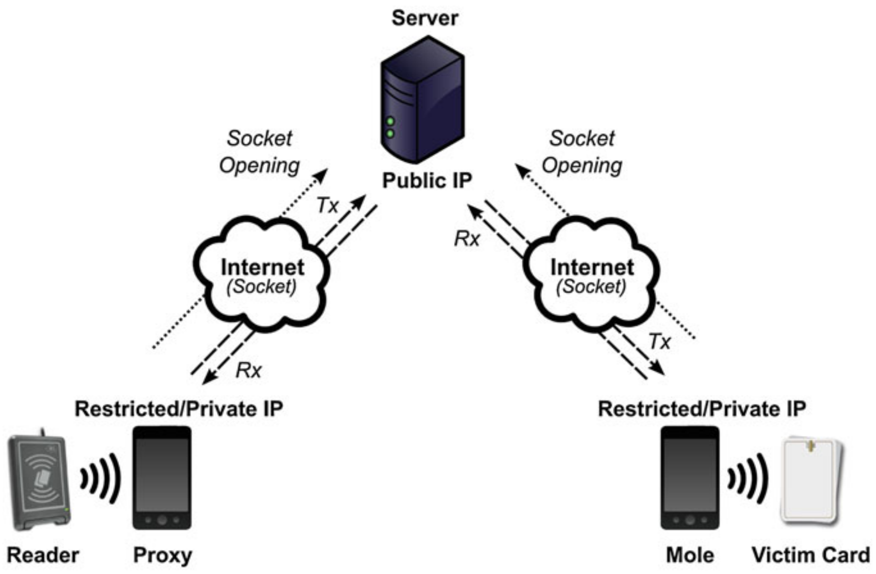
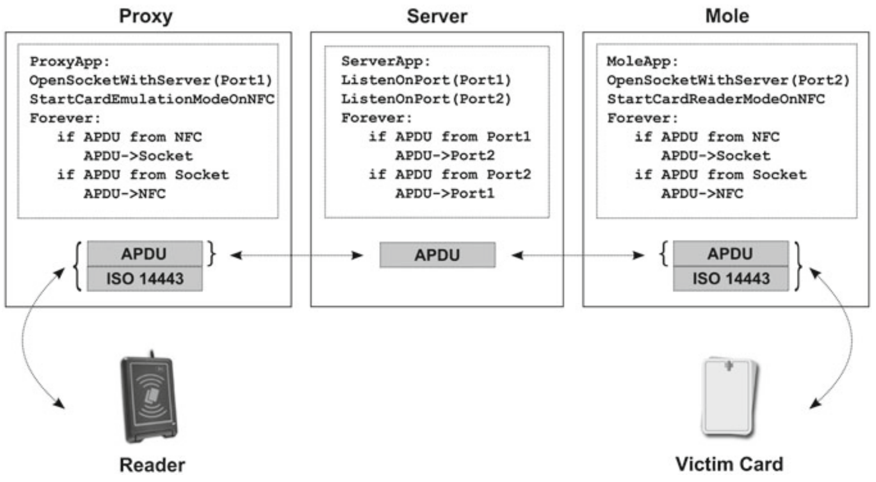


Fig. 2. Relay attack architecture based on mobile phones with an Internet connection.

the application starts the card emulation modality. The two applications should establish a TCP/IP connection between the devices and whenever an APDU reader command is received on the Proxy's NFC interface by the reader, the Proxy software forwards it towards the Mole, whose application re-transmits the received command through the NFC interface towards the victim card. The response provided by the card is managed by the two applications in reverse order until it is returned to the reader, so accomplishing a full command-response interaction.

We examined the IP assigned by a couple of mobile phone network operators and realized that the scheme, as proposed above, was not achievable: the address assigned to the devices was a public IP but without the possibility to accept incoming connections. This means that none of the two phones could open a TCP port on its IP waiting for the incoming connection from the other phone, as they were not reachable from the outside. Since only outgoing connections were allowed on the devices (e.g., for browser connections) we had to tune our attack architecture accordingly, as shown in Fig. 2. We added a Server connected to the Internet through a public IP address and running a simple forwarding application. The Server application opened two TCP ports for incoming connections, one for the Proxy and one for the Mole. When the Proxy and Mole applications were started, they opened a socket towards the Server on the respective dedicated port (since they were outgoing connections, we did not have any problems with them). The Server application forwarded the data received on a socket to the other, so at this stage a bi-directional Proxy-(Server)-Mole connection was in place for the attack. The developed architecture is quite general



```
ProxyApp:
OpenSocketWithServer(Port1)
StartCardEmulationModeOnNFC
Forever:
    if APDU from NFC
        APDU->Socket
    if APDU from Socket
        APDU->NFC
```

```
ServerApp:
ListenOnPort(Port1)
ListenOnPort(Port2)
Forever:
    if APDU from Port1
        APDU->Port2
    if APDU from Port2
        APDU->Port1
```

```
MoleApp:
OpenSocketWithServer(Port2)
StartCardReaderModeOnNFC
Forever:
    if APDU from NFC
        APDU->Socket
    if APDU from Socket
        APDU->NFC
```

Fig. 3. Relay attack architecture: software organization and APDU messages forwarding.

and so applicable even if a private IP has been assigned to the phones, as it could be in the case of mobile phone network operators offering a service in a NAT configuration mode, or if the phones are connected to the Internet behind a WiFi connection with a router.

For the attack, the Mole has to be in proximity of the victim card with an initialized ISO/IEC 14443 communication between them. The Proxy is put close to the reader and they establish an ISO/IEC 14443 communication, upon which the reader starts sending APDU commands. When an APDU command gets to the Proxy application through the NFC interface, the byte string representing the command is simply forwarded on the socket opened with the Server, which receives the APDU byte string and straightaway forwards it on the socket established with the Mole. When the Mole application receives the byte string, it is just re-transmitted to the victim card through the NFC interface, relying on the active ISO/IEC 14443 communication. Vice versa for the APDU responses from the card. The whole attack is represented in Fig. 3. The critical aspect of such solution is represented by the Reader-Proxy command-response communication, as there is the mentioned constraint on the maximum time allowed to return the response to the reader. Concerning this the Proxy receives the command and forwards it towards the victim card through the designed architecture, waiting for the response. If this round trip takes too long the Proxy is not able to return the response in time and the reader aborts the communication. We have evaluated the delay introduced by our architecture at the Proxy level, highlighting the time needed by the Proxy to get the victim card response when a command has been received by the reader and forwarded to the Server. It is important to point out that some time constraints have to be followed during the initialization procedure of the ISO/IEC 14443 communication as well. Nevertheless, they do not create any problems in this solution, as the the communication initialization is run locally, between reader and Proxy on one side, and between Mole and victim card on the other, so without the introduction of any additional delays.

4 Long Distance Relay Attack Experiment

To prove the effectiveness of the proposed attack architecture we decided to apply it to ePassports, which contain an ISO/IEC 14443 compliant contactless card adopting APDUs for commands/responses and some cryptographic authentication mechanisms during the communication, conducting some experiments on a geographical scale.

4.1 Target Contactless Card

Nowadays several countries issue ePassports [14,15], an electronic version of the traditional passport that embeds a contactless smart card in its cover, which relies on the stack presented in Sect. 2 for the communication. Several passport holder personal data can be stored in the card chip along with some biometrics, like facial and fingerprint images. Due to the importance of such a document and the sensitivity of the data stored, several cryptographic mechanisms have been adopted to secure the embedded smart card. Some of them are summarized in Fig. 4, which are the ones relevant for our experiments. For further details about ePassport security mechanisms please refer to the official specifications [14,15].

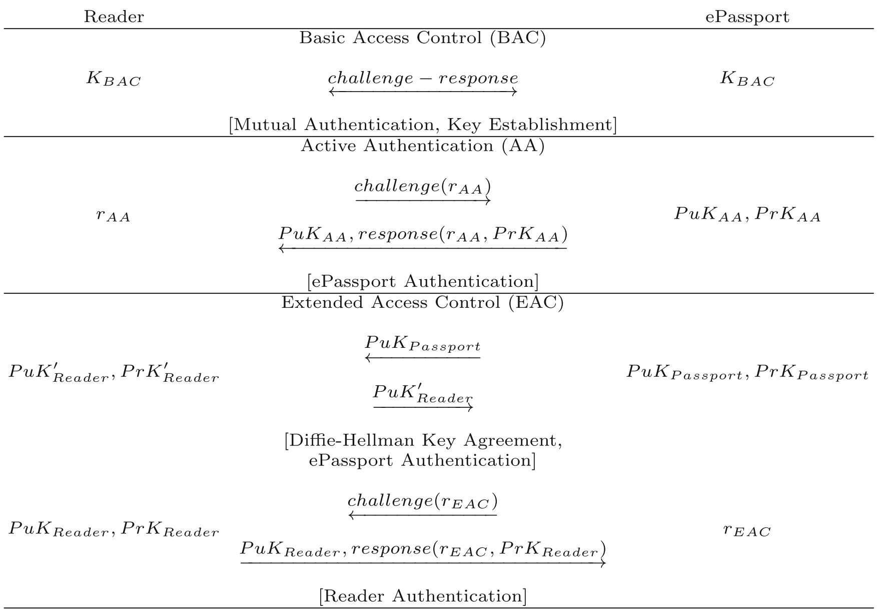

Reader
ePassport
Basic Access Control (BAC)
K_{BAC}
$challenge - response$
K_{BAC}
[Mutual Authentication, Key Establishment]
Active Authentication (AA)
$challenge(r_{AA})$
r_{AA}
$PuK_{AA}, response(r_{AA}, PrK_{AA})$
PuK_{AA}, PrK_{AA}
[ePassport Authentication]
Extended Access Control (EAC)
$PuK_{Passport}$
$PuK'_{Reader}, PrK'_{Reader}$
PuK'_{Reader}
$PuK_{Passport}, PrK_{Passport}$
[Diffie-Hellman Key Agreement, ePassport Authentication]
$challenge(r_{EAC})$
$PuK_{Reader}, PrK_{Reader}$
$PuK_{Reader}, response(r_{EAC}, PrK_{Reader})$
r_{EAC}
[Reader Authentication]

Fig. 4. Reader-ePassport authentication mechanisms. BAC is based on symmetric key cryptography whereas AA and EAC are based on public key cryptography.

To access an ePassport chip the reader has typically to run first the Basic Access Control with it: reader and chip share a couple of symmetric keys (K_{BAC}), which are used in a challenge-response protocol to mutually authenticate them and to establish a couple of session keys used to encrypt and authenticate the communication. The K_{BAC} keys are derived from the passport number, the passport expiry date and the passport holder date of birth through a deterministic process. Such data have to be manually inserted reader side, or automatically scanning the ePassport data page, to derive the K_{BAC} keys, which are typically already stored in the card's chip. The idea is to restrict the chip access only to the readers aware of the above-mentioned data (e.g., you allow an officer to read out your ePassport data giving him such information). After a successful BAC several personal data (e.g., name, surname, gender) and a facial image can be got from the chip.

The Active Authentication (AA) was designed to prevent chip cloning. The chip stores a public and private key pair (PuK_{AA}, PrK_{AA}) that are used in a challenge-response protocol. PrK_{AA} is securely stored and cannot be read from the outside, whereas PuK_{AA} is readable and digitally signed by the passport issuing country, with the reader that verifies such signature. The reader sends the challenge r_{AA} that the ePassport returns signed using PrK_{AA}: the reader verifies the signature using PuK_{AA} received from the chip. If the process is successful, the chip is authenticated and the reader is confident that an original ePassport has been put in its proximity. Indeed, even if the passport chip content has been copied on another card, the correct PrK_{AA} cannot be copied and the AA would fail.

The Extended Access Control (EAC) is basically divided in two parts and was introduced to further protect other sensitive data (e.g., fingerprints) that can be stored on the chip [16]. The first, called Chip Authentication, is used to run a Diffie-Hellman key agreement protocol between reader and ePassport and to authenticate the latter. The chip stores a public and private key pair ($PuK_{Passport}$, $PrK_{Passport}$), where again $PrK_{Passport}$ is securely stored and is not accessible from the outside, whereas $PuK_{Passport}$ is readable and digitally signed by the passport issuing country with such signature verified by the reader. An ephemeral public and private key pair (PuK'_{Reader}, PrK'_{Reader}) is generated by the reader. The two key pairs are used to run a Diffie-Hellman protocol and to agree on a common secret used for the encryption of the following communication. In such a way the chip is also implicitly authenticated, as if the wrong $PrK_{Passport}$ is used the communication fails, so this is an additional measure against chip cloning. The second part, called Terminal Authentication, is a challenge-response protocol to authenticate the reader, which stores the key pair (PuK_{Reader}, PrK_{Reader}), with PuK_{Reader} that is signed by the passport issuing country with such signature verified on the document chip. If the authentication is successful the chip grants the reader the access to the additional data present in its memory.

Not all the mechanisms are mandatory and their use can be conditioned by the presence of some data (e.g., EAC required if fingerprints are present), but on the other hand there are not any specifications preventing to used them all together (e.g., when EAC is required also AA can be added as further chip authentication measure). That being so, we decided to test our attack against an ePassport adopting all possible security mechanisms, so using the ICAO SDK Pro application [17] we burned on a contactless smart card an ePassport featured by BAC, AA, EAC, storing personal data, a facial picture and two fingerprint images. We remark that the prepared card was a fully compliant ePassport, featured by hardware and software that are typically used in commonly issued ePassports. In addition, it has to be pointed out that there are no in circulation many ePassports using AA and there are also currently problems to run the EAC protocol on ePassports issued by countries that adopt such solution, as it is not easy to get a reader equipped with a signed key pair to accomplish the second part of the protocol. Thanks to our prepared ePassport we were able to set up a reader station capable of running full EAC and AA procedures with it, installing the proper keys on the reader and on the chip, so giving us the possibility to test the relay attack against both mechanisms to check the chip authenticity. We also remark that the ePassport is just used as example of contactless smart card adopting mechanisms to prove its authenticity, but our attack is applicable in principle to all fully compliant ISO/IEC 14443 cards that use analogous security mechanisms.

4.2 Attack Set-Up and Results

To implement the attack we relied on an HTC One X with Android 4.1.1 as Mole and a Sony Xperia S with Android 4.0.4 as Proxy, installing the CynogenMod 9.1 firmware on the latter that had to emulate a card in front of our reader. As Server we used a desktop computer with an Intel i5 3.4GHz CPU, 8GB of RAM and Windows 8 64bit, connected to the Internet through a 6 Mbits ADSL connection.

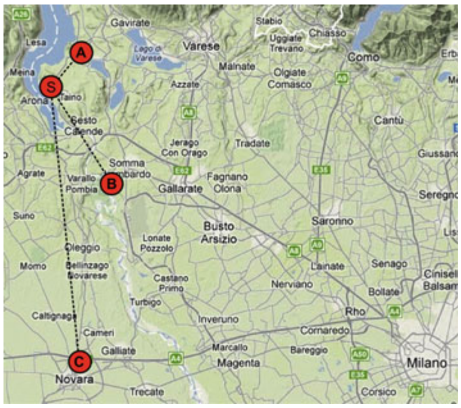


Fig. 5. Experiment area (Northern Italy): Proxy in A, Mole respectively in A, B and C, Server in S.

As reading system we adopted a laptop with an Intel i3 2.53GHz CPU, 4GB of RAM and Windows 7 64bit, plugging-in an ACS ACR122 USB contactless smart card reader and installing the Golden Reader Tool [18], a reference application for reading ePassports. The Server, Mole and Proxy applications were developed in Java. We remark that code examples were available for the implementation of the card emulation application [11], making our work even simpler.

We focused on four attack scenarios. In the first one, Server, Mole and Proxy were all connected to a router in WiFi in our labs and the WLAN was used for all Mole-Server and Proxy-Server communications. In the other three scenarios the devices where geographically deployed as depicted in Fig. 5: the Server was in S, the Proxy with the reading system was in A, while ePassport and Mole were placed respectively in A, B and C, achieving a distance in straight line of ∼15 Km for A-B and ∼42 Km for A-C. The phones were connected to the Internet through the data network offered by the mobile phone network operator, while the Server exploited its ADSL connection that offered a public IP, so the communications between phones and Server were conveyed through the Internet.

In all scenarios the Mole was put in proximity of the ePassport card while the Proxy was nearby our reader. In such conditions we started the Golden Reader Tool inserting the necessary data for a successful BAC with our ePassport. For all scenarios the application and the card were able to remotely interact, successfully running all the cryptographic security mechanisms of Fig. 4 and exchanging all the data stored in the chip. In particular the reader was always cheated successfully running AA and EAC protocols, thus making it believe that a genuine chip was in its proximity.

For all our experiments we measured the time that Mole and Proxy had to wait to get a response. In particular, the waiting time for the Mole was defined as the time elapsed between a command sent to the card and the received response, whereas for the Proxy it was the time elapsed between a command forwarded to the Server and the received relative response from it. For the second case, we highlight that the whole Proxy-Server-Mole-Card round trip of the messages contributed to the amount of time. Mole timings were similar for all experiment scenarios, since depending only on the Mole-Card interaction, so we grouped

Table 1. Response waiting time mean and standard deviation for Mole and Proxy.

Response Waiting Time Mole Side (ms)

Mean	Std.Dev.
144	94

Response Waiting Time Proxy Side (ms)

	Mean	Std.Dev.
Proxy and Mole in WLAN	252	125
Proxy and Mole in A	559	301
Proxy in A, Mole in B (∼ 15Km)	628	371
Proxy in A, Mole in C (∼ 42Km)	555	340

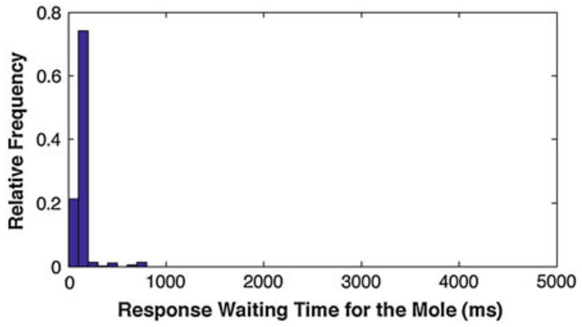


Fig. 6. Response waiting time frequency distribution Mole side (Relative Frequency: number of occurrences of a waiting time normalized by the total number of observed waiting times).

them all together, while Proxy timings were affected by the adopted communication network and its relative conditions. The presented data are derived from ∼1000 command-response interactions between reader and card per scenario, running several successful ePassport readings.

The observed Mole waiting times are presented in the higher part of Table 1 and in Fig. 6. It is evident that the times are always short enough to allow a successful phone-card interaction considering the maximum waiting time that can be requested by the card. In particular such interactions were always successful and we did not investigate on the value set for the FWI parameter. We suppose that the few higher times on the right of the peak shown in Fig. 6 were due to the internal chip computations for the cryptographic authentication operations required by the reader (i.e., AA and EAC).

In the lower part of Table 1 and in Fig. 7 we present the Proxy waiting times. On average, the times observed in all scenarios tend to allow a successful Proxy-Reader interaction. Indeed, the waiting time at the reader level for a transmitted command is given by the Proxy waiting time, plus the transmission times of the command and response messages exchanged with the Proxy itself through its NFC interface. As the time added by a reader-Proxy message exchange is roughly in the same order of magnitude of the mean Mole waiting time presented above, it becomes clear as the reader is able to receive the command response within the maximum waiting time allowed in a contactless card communication, thus making the attack successful. Note that we did not set any waiting time parameters in our Proxy application, but we just relied on the default FWI parameter set by the underlying library and firmware installed on the phone to use the NFC interface and emulate a card. To have an insight about its value, we conducted some tests artificially delaying in the Proxy software the responses returned to the reader until the communication failed. As the reader aborted the communication for delays roughly equal to or greater than 5 s, we realized that the FWI parameter was set by default to have the maximum waiting time, so making the attack implementation even easier.

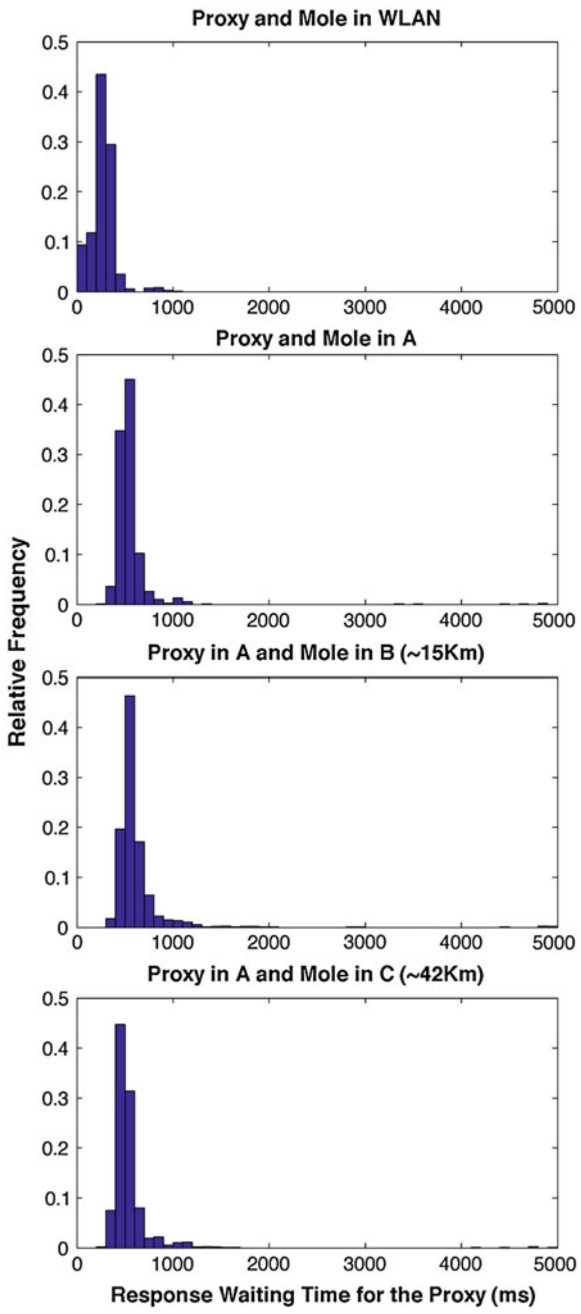


Fig. 7. Response waiting time frequency distributions Proxy side (Relative Frequency: number of occurrences of a waiting time normalized by the total number of observed waiting times).

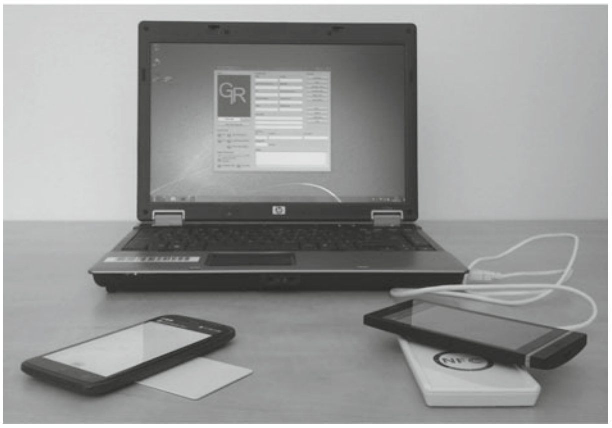


Fig. 8. Off-the-shelf equipment used in the relay attack.

As evident by the shown results, and as predictable, we achieved a better performance when the communication was conveyed through the WLAN, with a Proxy waiting time that was on average half of the time got through a mobile phone data network and in general always below 1 s, so granting a perfect card-reader interaction. Nevertheless, the mean waiting time for a communication based on the mobile phone network was around 500/600 ms, with most of the waiting times that were below 1 s, so resulting in an efficient architecture for successful relay attacks. According to the charts of Fig. 7 there are also few waiting times of some seconds. We analyzed such times and we found out that they were always associated with the first command-response pair of an ePassport reading. After several tests we argue that it was due to a switching of the mobile phone communication technology adopted by the two devices. If no data were transmitted or received by the phones, they were basically connected to the mobile phone network through an UMTS base connection, but as soon as a data exchange was ongoing, the phones switched to an HSPA connection, an enhanced UMTS communication featured by an higher data rate. The switching was pointed out on their displays, it happened on both devices at the beginning of the ePassport reading and we noted that it took a bit of time. To support our assumption we repeated for a while the following tests: before starting an ePassport reading we launched the download of a couple of applications on the two phones, so that they switched to the HSPA connection due to the data exchange, and then we ran the ePassport reading. In such cases the first command-response waiting time was in line with the other waiting times. However this is a hypothesis, as the delay could be also due to some peripheral idle-wake up procedures in place for power saving purposes on the devices, or some TCP mechanisms, so in general further investigations would be needed concerning these higher waiting times. We have to point out as from time to time these long times caused a failure at the beginning of the reading process, but if no errors occurred during the first

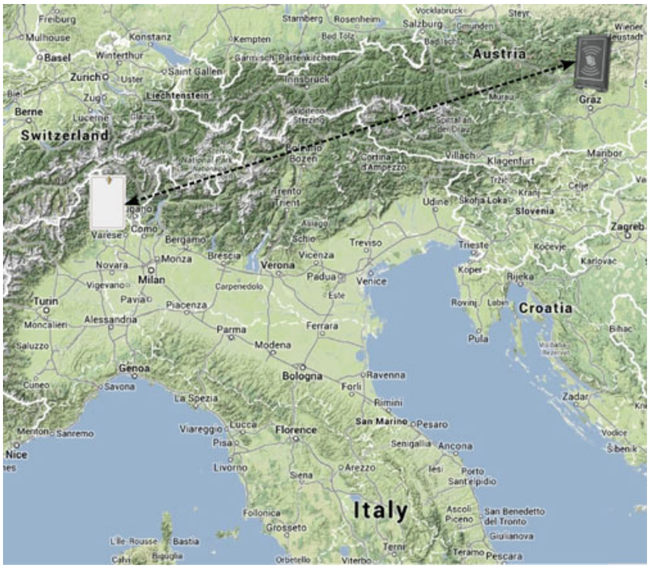


Fig. 9. RFIDsec 2013 demo: relay attack over ∼541 Km, with the card in Italy and the reader in Austria.

command-response, then the reading was conducted until the end without any problems.

We note that the distance does not seem to play an important role, since the three scenarios based on the mobile phone data network are featured by similar results. In addition, the scenario A-B, featured by a shorter distance compared to the case A-C, presents a slightly higher waiting time mean and standard deviation, aspect that can be probably attributed to the different mobile phone signal quality in various places, so making this characteristic more important than the distance.

The attack was also presented through a live demo during the RFIDsec 2013 event. The Server and the Mole were respectively located in S and A as shown by Fig. 5. The Proxy and the reading station were in Graz (Austria), location of the conference. For this specific case Mole and Proxy were connected to the Internet through a WiFi router, solution allowed by our architecture: the Mole used the WiFi network of our labs while the Proxy was connected to the WiFi offered by the hotel that hosted the conference. The Mole and the card located in Italy were shown to the audience through a Skype video session. The communication between card and reader was successfully relayed achieving a distance in straight line of ∼541 Km (Fig. 9).

5 Discussion

The attack, as we have been able to implement it, could raise real threats for the end user. It is entirely based on cheap and off-the-shelf tools, the code that has to be developed is simple and some templates and examples are available, it is not restricted by fixed positions with Mole and Proxy that can be in whatever place covered by a mobile phone signal and it is general as designed to relay card commands and responses without any links to the specific card application (our specific application to ePassports is just an example). In addition the presented architecture can be easily enhanced to have, not a single, but several Moles controlled by a Proxy. In such a case an attacker with its Proxy could potentially exploit a large set of contactless cards at his service. Concerning this, it has to be considered that many people keep their phone close to their wallet (e.g., in a pocket, in a purse), so once the Mole application has been installed on the victim's phone it could start polling the different cards present in the nearby wallet (e.g., payment cards, ID documents, access control tokens), identifying them and returning the information to the Proxy, with the attacker who dynamically knows which "services" (i.e., which cards) are offered by that specific Mole. We remark that the proposed architecture is generic, so the attack would be achievable also when the phones switch their Internet connection, with several scenarios that are possible (e.g., Mole connected to a WiFi and Proxy to a mobile phone network). For the installation of the Mole application on the victim's phones there are different possible ways [19], as for instance through social engineering channels, exploiting Bluetooth vulnerabilities or distributing the Mole software through phone applications repositories (e.g., coupling the Mole application with a game). Note that the more complex firmware replacement operation has to be done on the Proxy phone only, which is owned by the attacker.

We point out that other fraudulent scenarios are possible exploiting the proposed attack architecture. For instance the Mole could be held by an accomplice who using an equipment to extend the contactless card reading range of his device [20] approaches victims with contactless cards. A similar architecture could be exploited to use a contactless card in different places at the same time (single Mole nearby the card with two or more Proxies), causing confusion on the real position of a cardholder at a specific time when the relative card transaction logs are examined (for instance to create a possible alibi for a trial in court).

We remark that the effectiveness of the proposed attack architecture have been proved for an ISO/IEC 14443 compliant contactless card with the communication architecture presented in Sect. 2, but further experiments should be carried out to verify the possibility to relay the communication of cards that rely on a different communication solution. We have also to point out, even if we did not need it, that a Waiting Time Extension (WTX) command is available for ISO/IEC 14443 compliant communications [8]: it can be used by the chip to request further time, other than the time specified by the FWI, to return the response to the reader. Such commands could be useful to run relay attacks on networks or architectures featured by a lower performance than the one we have adopted.

The proposed attack architecture confirms that it is no longer possible to assume that for sensitive operations a contactless card has to be nearby a reader.

In addition, different measures put in place to assure that a genuine card is in front of a reader are not effective against such an attack. For instance in our experiments we have adopted two cryptographic protocols to verify the authenticity of the card, the AA and EAC mechanisms offered by the ePassport, but both were simply remotely performed with a positive result. A mechanism like BAC represents a good deterrent: to run our experiments we inserted on purpose the ePassport BAC information on the reader, but if it was not aware of such data the reading could not be carried out. Nevertheless such information is static, printed in the passport booklet and can be present on some papers (e.g., boarding cards, hotel forms), so it can be got for a target document (e.g., a hotel receptionist could collect customer passport data and then attempt the installation of the Mole software through social engineering techniques). Also the adoption of a PIN represents a good way to protect our cards, but it is usually not used to foster the fast transactions allowed by these contactless chips, as for instance in the contactless credit card case [7]. On the other hand, if a PIN authentication is in place, a denial of service attack could be performed against the victim remotely blocking his card (usually after a defined number of wrong PIN attempts the card is blocked [7]). In addition, in a future vision, if security mechanisms are used to protect contactless cards, the Moles could be instructed to launch autonomous attacks against their cards to unlock them, notifying the Proxy at the end of the process.

In order to protect such cards some changes in the current adopted standards are needed, maybe adding some constraints on the maximum allowed waiting time, measuring the command-response time (it seems that a similar custom solution is adopted in Mifare Plus cards [21]) or evaluating the possible use of distance-bounding protocols [22]. Nowadays, the only effective countermeasure for ISO/IEC 14443 fully compliant cards is represented by card shielding covers that acting as a Faraday cage hinder the communication with the embedded contactless chip.

6 Conclusions

In this paper we present a cheap and practical implementation of a long distance relay attack against contactless smart cards. The attack is based on two NFC mobile phones, which through a proper architecture and exploiting a mobile phone data network connection, are able to set up a communication link between a card and a reader that are physically far away. We have proved the effectiveness of the proposed solution running some experiments on a geographical scale, relaying messages over kilometers and showing that basically there are not any logistic constraints for card and reader.

References

1. Kfir, Z., Wool, A.: Picking virtual pockets using relay attacks on contactless smartcard. In: Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm), pp. 47–58 (2005)

2. Hancke, G.P.: A practical relay attack on ISO 14443 proximity cards. Technical report, University of Cambridge Computer Laboratory, pp. 1–13 (2005)
3. Thevenon, P., Savry, O., Tedjini, S.: On the weakness of contactless systems under relay attacks. In: Proceedings of the 19th International Conference on Software, Telecommunications and Computer Networks (SoftCOM), pp. 1–5 (2011)
4. Issovits, W., Hutter, M.: Weaknesses of the ISO/IEC 14443 protocol regarding relay attacks. In: Proceedings of the International Conference on RFID-Technologies and Applications (RFID-TA), pp. 335–342 (2011)
5. WeiB, M.: Performing relay attacks on ISO 14443 contactless smart cards using NFC mobile equipment. Master thesis, Der Technischen Universitat Munchen, Germany (2010)
6. Francis, L., Hancke, G., Mayes, K., Markantonakis, K.: Practical relay attack on contactless transactions by using NFC mobile phones. In: Proceedings of the Workshop on RFID and IoT Security (RFIDsec 2012 Asia) (2012)
7. Emms, M., Arief, B., Defty, T., Hannon, J., Hao, F., van Moorsel, A.: The dangers of verify PIN on contactless cards. Newcastle University, Technical Report Series, No. CS-TR-1332, pp. 1–10 (2012)
8. ISO/IEC 14443: Identification cards - Contactless Integrated Circuit Cards - Proximity Cards (2011)
9. ISO/IEC 7816–4: Identification Cards - Integrated Circuit Cards - Part 4: Organization, Security and Commands for Interchange (2005)
10. ISO/IEC 21481: Information technology - Telecommunications and Information Exchange Between Systems - Near Field Communication Interface and Protocol -2 (NFCIP-2) (2005)
11. Elenkov, N.: Emulating a PKI Smart Card with CyanogenMod 9.1 (2012). http://nelenkov.blogspot.it/2012/10/emulating-pki-smart-card-with-cm91.html
12. CyanogenMod, Ver. 9.1. http://www.cyanogenmod.org/ (2013)
13. Roland, M.: Software card emulation in NFC-enabled mobile phones: great advantage or security nightmare? In: Fourth International Workshop on Security and Privacy in Spontaneous Interaction and Mobile Phone Use, pp. 1–6 (2012)
14. International Civil Aviation Organization: Machine Readable Travel Documents, Part 1, vol. 1, 6th edn (2006)
15. International Civil Aviation Organization: Machine Readable Travel Documents, Part 1, vol. 2, 6th edn (2006)
16. BSI: Advanced Security Mechanisms for Machine Readable Travel Documents - Extended Access Control (EAC), Password Authenticated Connection Establishment (PACE) and Restricted Identification (RI). Ver. 2.05 (2010)
17. ICAO SDK Pro, MaskTech GmbH (2008)
18. Golden Reader Tool, Ver. 2.9.4 (2009). https://www.bsi.bund.de/EN/Topics/ElectrIDDocuments/Projects/projectsGRT/GRT_node.html
19. Dunham, K.: Mobile Malware Attacks and Defense. Syngress, Burlington (2009)
20. Kirschenbaum, I., Wool, A.: How to build a low-cost, extended-range RFID skimmer. In: Proceedings of the 15th USENIX Security, Symposium, pp. 43–57 (2006)
21. MF1PLUSx0y1 - Mainstream Contactless Smart Card IC for Fast and Easy Solution Development, Product short data sheet, Rev. 3.2, NXP (2011)
22. Hancke, G.P., Kuhn, M.G.: An RFID distance bounding protocol. In: Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm), pp. 67–73 (2005)

On the Security of Two RFID Mutual Authentication Protocols

Seyed Farhad Aghili[1], Nasour Bagheri[1](✉), Praveen Gauravaram[2], Masoumeh Safkhani[3], and Somitra Kumar Sanadhya[4]

[1] Electrical Engineering Department, Shahid Rajaee Teacher Training University, Tehran, Iran
NBagheri@srttu.edu

[2] Innovation Labs Hyderabad, Tata Consultancy Services Limited, Hyderabad, India
P.Gauravaram@tcs.com

[3] Electrical Engineering Department, Iran University of Science and Technology, Tehran, Iran
M_Safkhani@iust.ac.ir

[4] Indraprastha Institute of Information Technology, Delhi, India
Somitra@iiitd.ac.in

Abstract. In this paper, the security of two recent RFID mutual authentication protocols are investigated. The first protocol is a scheme proposed by Huang et al. [7] and the second one by Huang, Lin and Li [6]. We show that these two protocols have several weaknesses. In Huang et al.'s scheme, an adversary can determine the 32-bit secret *Access* password with a probability of 2^{-2}, and in Huang-Lin-Li scheme, a passive adversary can recognize a target tag with a success probability of $1-2^{-4}$ and an active adversary can determine all 32 bits of *Access* password with success probability of 2^{-4}. The computational complexity of these attacks is negligible.

Keywords: RFID · EPC Class-1 Generation-2 · PadGen function

1 Introduction

Radio frequency identification (RFID) uses radio frequency signals to identify objects or people automatically. Typically, the main components of an RFID system are an RFID tag, RFID reader and a back-end server [14]. The main function of an RFID system is identification and authentication. Hence most of the RFID applications need to provide authentication between a tag and a reader. Authentication is a process in which one party is assured of the identity of the another party by obtaining the required evidences, which is done in a corroborative manner. In our case these parties are the Tag and Reader/back-end database. A secure authentication protocol is expected to resist against the attacks in the scenarios such as rogue scanning, replay attack and tag counterfeiting or cloning.

On the other hand, several interconnected standards exist for RFID systems. Among them, ISO [8] and Electronic Product Code (EPC) global [5] have played the main role. The EPC Class-1 Generation-2 (C1 G2) is a universal standard for low-cost passive RFID tags. This group of tags is also covered by ISO 18000-6C standard.

EPC-C1 G2 specifies that any RFID tag compliant with this standard should contain two 32-bit passwords denoted by the access password and the kill password respectively. The access password is used to authenticate the reader that wish to access information inside the tag and control access to the information. The kill password is generally used to disable the tag. A killed tag is rendered in silence thereafter and does not respond to any query from any reader. EPC-C1 G2 standard proposes a simple authentication protocol that allows a tag to authenticate a reader. This protocol attempts to protect the access password by using a simple form of masking before transmission over a wireless channel. This masking which is known as pad generation ($PadGen$) is a simple bitwise XOR. However, a passive adversary monitoring the exchanged messages between the reader and the tag can retrieve this sensitive information easily [1,13]. These results have motivated researchers to try to propose EPC-compliant authentication protocols to improve its security level. However, the main difficulty in providing a mutual authentication protocol for RFID systems with passive tags is the very limited storage and computational capabilities of EPC- C1 G2 tags that significantly limits their support for conventional cryptographic primitives such as AES. To provide the desired security of the tags that support this standard, several mutual authentication protocols [2–4,11] were proposed. In this direction, Konidala et al. have proposed an RFID mutual authentication protocol to solve ISO 18000-6C protocol weaknesses [9]. However, the designed protocol is known to be flawed and the adversary can retrieve most of the secret passwords' bits efficiently [12]. To solve Konidala et al. protocol's weakness two novel protocols described in [6,7] have recently been proposed. In this paper, these protocols are denoted by HYCLT and HLL respectively. These protocols do not use any standard cryptographic primitives and attempt to provide the desired security by simple logical operations. Note that Ma et al. [10] have shown that any RFID protocol without using PRF is subject to some kind of tag tracing attacks. We show that this is indeed the case for the current protocols both of which do not utilize a PRF. We investigate the security level of these protocols and present practical attacks to retrieve tag's secret parameters.

Our contribution: Any tag in HYCLT and HLL protocols have two 32-bit passwords called *Kill* password and *Access* password respectively. We investigate the security of protocols against secret disclosure attack and show that an adversary can determine whole *Access* password of HYCLT with a probability of 2^{-2} at a cost of a single query to the target tag. We also analyze the security of HLL protocol and present several tag recognizing attacks against it. In the presented attacks, given a tag, the adversary can recognize whether it is the target tag with the probability of $1 - 2^{-4}$. In addition, we show that a man in the middle adversary can determine whole *Access* and *Kill* passwords with success probability of 2^{-4} for negligible complexity.

Paper organization: The rest of the paper is organized as follows: In Sect. 2 we present HYCLT protocol description and discuss its security. In Sect. 3 we describe the HLL protocol and investigate its security. In Sect. 4 we conclude the paper.

2 HYCLT Mutual Authentication Protocol

Konidala et al. [9] have proposed an RFID mutual authentication protocol to solve ISO 18000-6C protocol weaknesses [9] by using a special PadGen function to mask tag's *Access* password $Apwd = Apwd_L \| Apwd_M$ before the data is transmitted. However, Konidala et al. protocol suffers from correlation attack [12]. To solve Konidala et al. protocol's weakness, Huang et al. have proposed an improved version based on a different PadGen and also successfully demonstrated the FPGA hardware implementation of their proposed mutual authentication protocol [7]. We denote this protocol by HYCLT. The notation used in the paper are depicted in Table 1.

The PadGen function proposed in HYCLT accepts a 32-bit value and two 16-bit values as input and outputs 16 bits. Given $\mathcal{X} \in \{0,1\}^{32}$, we can represent it as $\mathcal{X} = \mathcal{X}|_0 \mathcal{X}|_1 \dots \mathcal{X}|_{31}$, where $\mathcal{X}|_i \in \{0,1\}$, and given 16-bit values $\mathcal{Y} \in \{0,1\}^{16}$ and $\mathcal{Z} \in \{0,1\}^{16}$, they can be represented as $\mathcal{Y} = d_{\mathcal{Y}1} d_{\mathcal{Y}2} d_{\mathcal{Y}3} d_{\mathcal{Y}4}$ and $\mathcal{Z} = d_{\mathcal{Z}1} d_{\mathcal{Z}2} d_{\mathcal{Z}3} d_{\mathcal{Z}4}$, where $d_{\mathcal{Z}i} \in \{0,1,\dots,15\}$, $i \in \{1,2,3,4\}$ and used as base 10(decimal) representation of a four-bit binary string. For example, $\mathcal{Z} =$ 1101 0110 1000 1001 can be represented as $\mathcal{Z} =$ 13 06 08 09 which means that $d_{\mathcal{Z}1} = 13, d_{\mathcal{Z}2} = 06, d_{\mathcal{Z}3} = 08, d_{\mathcal{Z}4} = 09$. Similarly, one can represent $\mathcal{Y}$ and $\mathcal{Z}$ as $\mathcal{Y} = h_{\mathcal{Y}1} h_{\mathcal{Y}2} h_{\mathcal{Y}3} h_{\mathcal{Y}4}$ and $\mathcal{Z} = h_{\mathcal{Z}1} h_{\mathcal{Z}2} h_{\mathcal{Z}3} h_{\mathcal{Z}4}$, where $h_{\mathcal{Z}i} \in \{0,1,\dots,F\}$, $i \in \{1,2,3,4\}$ and used as base hexadecimal (base 16) representation of a four-bit binary string. For example, $\mathcal{Z} =$ 1101 0110 1000 1001 can be represented as

Table 1. Notation

Notation	Description
R_i	RFID reader i
T_i	RFID tag i
Req_R	Reader request
R_{Tx}	Random numbers generated by the tag
R_{Mx}	Random numbers generated by the server
EPC	Electronic product code
$Apwd$	*Access* password
$Apwd_L$	16 least significant bits of $Apwd$
$Apwd_M$	16 most significant bits of $Apwd$
$Kpwd$	*Kill* password
$X\|_i$	ith bit of string X
$\mathcal{X}\|_{m\sim n}$	A fraction of $\mathcal{X}$ from the mth bit to the nth bit
d_{Xi}	Decimal equivalent(base 10) of the ith 4-bit of string X
h_{Xi}	Hexadecimal(base 16) equivalent of the ith 4-bit of string X
$\oplus$	Exclusive or operation
$\|\|$	Concatenation operation

$\mathcal{Z} = D\ 6\ 8\ 9$ which means that $h_{\mathcal{Z}1} = D, h_{\mathcal{Z}2} = 6, h_{\mathcal{Z}3} = 8, h_{\mathcal{Z}4} = 9$. Given these definitions $PadGen(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$ is calculated as follows:

$$\begin{aligned} PadGen(\mathcal{X}, \mathcal{Y}, \mathcal{Z}) = {} & \mathcal{X}|_{d_{\mathcal{Y}1}}\mathcal{X}|_{d_{\mathcal{Y}1}+16}\mathcal{X}|_{d_{\mathcal{Y}2}}\mathcal{X}|_{d_{\mathcal{Y}2}+16}\|\mathcal{X}|_{d_{\mathcal{Z}1}}\mathcal{X}|_{d_{\mathcal{Z}1}+16}\mathcal{X}|_{d_{\mathcal{Z}2}}\mathcal{X}|_{d_{\mathcal{Z}2}+16}\| \\ & \mathcal{X}|_{d_{\mathcal{Y}3}}\mathcal{X}|_{d_{\mathcal{Y}3}+16}\mathcal{X}|_{d_{\mathcal{Y}4}}\mathcal{X}|_{d_{\mathcal{Y}4}+16}\|\mathcal{X}|_{d_{\mathcal{Z}3}}\mathcal{X}|_{d_{\mathcal{Z}3}+16}\mathcal{X}|_{d_{\mathcal{Z}4}}\mathcal{X}|_{d_{\mathcal{Z}4}+16} \end{aligned}$$

For example assume that:

$$\begin{aligned} \mathcal{X} &= 1001\ 1111\ 0011\ 1011\ 0000\ 0011\ 1100\ 0101 \\ \mathcal{Y} &= 0111\ 0100\ 0110\ 1011 \\ \mathcal{Z} &= 1101\ 0110\ 1000\ 1001 \end{aligned}$$

then $PadGen(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$ is calculated as follows:

$$\begin{aligned} PadGen(\mathcal{X}, \mathcal{Y}, \mathcal{Z}) &= \mathcal{X}|_7\mathcal{X}|_{7+16}\mathcal{X}|_4\mathcal{X}|_{4+16}\|\mathcal{X}|_{13}\mathcal{X}|_{13+16}\mathcal{X}|_6\mathcal{X}|_{6+16}\| \\ & \quad \mathcal{X}|_6\mathcal{X}|_{6+16}\mathcal{X}|_{11}\mathcal{X}|_{11+16}\|\mathcal{X}|_8\mathcal{X}|_{8+16}\mathcal{X}|_9\mathcal{X}|_{9+16} \\ &= 1110\ 0111\ 1110\ 0101 \end{aligned}$$

where:

$$\begin{aligned} \mathcal{X} = 1001\ & \underbrace{1}_{\mathcal{X}|_4}\ 1\ \underbrace{1}_{\mathcal{X}|_6}\underbrace{1}_{\mathcal{X}|_7}\ \underbrace{0}_{\mathcal{X}|_8}\underbrace{0}_{\mathcal{X}|_9}\ 1\ \underbrace{1}_{\mathcal{X}|_{11}}\ 1\ \underbrace{0}_{\mathcal{X}|_{13}} \\ & 11\ 0000\ \underbrace{0}_{\mathcal{X}|_{20}}\ 0\ \underbrace{1}_{\mathcal{X}|_{22}}\underbrace{1}_{\mathcal{X}|_{23}}\ \underbrace{1}_{\mathcal{X}|_{24}}\underbrace{1}_{\mathcal{X}|_{25}}\ 0\ \underbrace{0}_{\mathcal{X}|_{27}}\ 0\ \underbrace{1}_{\mathcal{X}|_{29}}\ 01 \end{aligned}$$

In HYCLT protocol, the tag and the server use the PadGen function to generate four masking values denoted by PAD_1, PAD_2, PAD_3, and, PAD_4 respectively. Let us to represent the 32-bit *Access* password $Apwd$ and the 32-bit *Kill* password $Kpwd$ as $Apwd = a|_0a|_1a|_2a|_3 \ldots a|_{31}$ and $Kpwd = k|_0k|_1k|_2k|_3 \ldots k|_{31}$ respectively where $a|_i \in \{0, 1\}$ and $k|_i \in \{0, 1\}$. Given 16-bit random numbers R_{Tx} and R_{Mx}, for $x \in \{1, 2, 3, 4\}$, they can be represented as $R_{Tx} = d_{R_{Tx}1}d_{R_{Tx}2}d_{R_{Tx}3}d_{R_{Tx}4}$ and $R_{Mx} = d_{R_{Mx}1}d_{R_{Mx}2}d_{R_{Mx}3}d_{R_{Mx}4}$.

The PadGen function of HYCLT protocol is used to compute masking values PAD_x, for $x \in \{1, 2, 3, 4\}$, as follows:

$$\begin{aligned} R_{Vx} &= PadGen(APwd, R_{Tx}, R_{Mx}) \\ &= a|_{d_{R_{Tx}1}}a|_{d_{R_{Tx}1}+16}a|_{d_{R_{Tx}2}}a|_{d_{R_{Tx}2}+16}\|a|_{d_{R_{Mx}1}}a|_{d_{R_{Mx}1}+16}a|_{d_{R_{Mx}2}}a|_{d_{R_{Mx}2}+16}\| \\ & \quad a|_{d_{R_{Tx}3}}a|_{d_{R_{Tx}3}+16}a|_{d_{R_{Tx}4}}a|_{d_{R_{Tx}4}+16}\|a|_{d_{R_{Mx}3}}a|_{d_{R_{Mx}3}+16}a|_{d_{R_{Mx}4}}a|_{d_{R_{Mx}4}+16} \\ &= d_{R_{Vx}1}d_{R_{Vx}2}d_{R_{Vx}3}d_{R_{Vx}4} \end{aligned}$$

and

$$\begin{aligned} PAD_x &= PadGen(Kpwd, R_{Vx}, R_{Tx}) \\ &= k|_{d_{R_{Vx}1}}k|_{d_{R_{Vx}1}+16}k|_{d_{R_{Vx}2}}k|_{d_{R_{Vx}2}+16}\|k|_{d_{R_{Tx}1}}k|_{d_{R_{Tx}1}+16}k|_{d_{R_{Tx}2}}k|_{d_{R_{Tx}2}+16}\| \\ & \quad k|_{d_{R_{Vx}3}}k|_{d_{R_{Vx}3}+16}k|_{d_{R_{Vx}4}}k|_{d_{R_{Vx}4}+16}\|k|_{d_{R_{Tx}3}}k|_{d_{R_{Tx}3}+16}k|_{d_{R_{Tx}4}}k|_{d_{R_{Tx}4}+16} \\ &= h_{PAD_x1}h_{PAD_x2}h_{PAD_x3}h_{PAD_x4} \end{aligned}$$

where R_{Vx} is a temporary variable. For example, PAD_1 is calculated as follows:

$$\begin{aligned} R_{V1} &= PadGen(APwd, R_{T1}, R_{M1}) \\ &= a|_{d_{R_{T1}1}} a|_{d_{R_{T1}1}+16} a|_{d_{R_{T1}2}} a|_{d_{R_{T1}2}+16} \| a|_{d_{R_{M1}1}} a|_{d_{R_{M1}1}+16} a|_{d_{R_{M1}2}} a|_{d_{R_{M1}2}+16} \| \\ &\quad a|_{d_{R_{T1}3}} a|_{d_{R_{T1}3}+16} a|_{d_{R_{T1}4}} a|_{d_{R_{T1}4}+16} \| a|_{d_{R_{M1}3}} a|_{d_{R_{M1}3}+16} a|_{d_{R_{M1}4}} a|_{d_{R_{M1}4}+16} \\ &= d_{R_{V1}1} d_{R_{V1}2} d_{R_{V1}3} d_{R_{V1}4} \end{aligned}$$

and

$$\begin{aligned} PAD_1 &= PadGen(Kpwd, R_{V1}, R_{T1}) \\ &= k|_{d_{R_{V1}1}} k|_{d_{R_{V1}1}+16} k|_{d_{R_{V1}2}} k|_{d_{R_{V1}2}+16} \| k|_{d_{R_{T1}1}} k|_{d_{R_{T1}1}+16} k|_{d_{R_{T1}2}} k|_{d_{R_{T1}2}+16} \| \\ &\quad k|_{d_{R_{V1}3}} k|_{d_{R_{V1}3}+16} k|_{d_{R_{V1}4}} k|_{d_{R_{V1}4}+16} \| k|_{d_{R_{T1}3}} k|_{d_{R_{T1}3}+16} k|_{d_{R_{T1}4}} k|_{d_{R_{T1}4}+16} \\ &= h_{PAD_1 1} h_{PAD_1 2} h_{PAD_1 3} h_{PAD_1 4} \end{aligned}$$

In this version of PadGen function, which is known as the simple version, 8 bits out of 16 bits of the resulted PAD_x are decided by R_{Tx}, i.e., the bits that are used to determine $h_{PAD_1 2}$ and $h_{PAD_1 4}$. To provide a better security, HYCLT also introduces a more complex approach to manipulate R_{Tx} and R_{Mx} on the output of PAD_x. Given these definitions and $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ the complex version of PadGen is calculated as follows:

$$\begin{aligned} PadGen(\mathcal{X}, \mathcal{Y}, \mathcal{Z}) = {} & \mathcal{X}|_{d_{\mathcal{Y}1}+d_{\mathcal{Z}1}} \mathcal{X}|_{d_{\mathcal{Y}1}+d_{\mathcal{Z}2}} \mathcal{X}|_{d_{\mathcal{Y}1}+d_{\mathcal{Z}3}} \mathcal{X}|_{d_{\mathcal{Y}1}+d_{\mathcal{Z}4}} \| \mathcal{X}|_{d_{\mathcal{Y}2}+d_{\mathcal{Z}1}} \mathcal{X}|_{d_{\mathcal{Y}2}+d_{\mathcal{Z}2}} \\ & \mathcal{X}|_{d_{\mathcal{Y}2}+d_{\mathcal{Z}3}} \mathcal{X}|_{d_{\mathcal{Y}2}+d_{\mathcal{Z}4}} \| \mathcal{X}|_{d_{\mathcal{Y}3}+d_{\mathcal{Z}1}} \mathcal{X}|_{d_{\mathcal{Y}3}+d_{\mathcal{Z}2}} \mathcal{X}|_{d_{\mathcal{Y}3}+d_{\mathcal{Z}3}} \mathcal{X}|_{d_{\mathcal{Y}3}+d_{\mathcal{Z}4}} \\ & \| \mathcal{X}|_{d_{\mathcal{Y}4}+d_{\mathcal{Z}1}} \mathcal{X}|_{d_{\mathcal{Y}4}+d_{\mathcal{Z}2}} \mathcal{X}|_{d_{\mathcal{Y}4}+d_{\mathcal{Z}3}} \mathcal{X}|_{d_{\mathcal{Y}4}+d_{\mathcal{Z}4}} \end{aligned}$$

For example assume that

$$\begin{aligned} \mathcal{X} &= 1001\ 1111\ 0011\ 1011\ 0000\ 0011\ 1100\ 0101 \\ \mathcal{Y} &= 0111\ 0100\ 0110\ 1011 \\ \mathcal{Z} &= 1101\ 0110\ 1000\ 1001 \end{aligned}$$

then $PadGen(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$ is calculated as follows:

$$\begin{aligned} PadGen(\mathcal{X}, \mathcal{Y}, \mathcal{Z}) &= \mathcal{X}|_{7+13} \mathcal{X}|_{7+6} \mathcal{X}|_{7+8} \mathcal{X}|_{7+9} \| \mathcal{X}|_{4+13} \mathcal{X}|_{4+6} \mathcal{X}|_{4+8} \mathcal{X}|_{4+9} \| \\ &\quad \mathcal{X}|_{6+13} \mathcal{X}|_{6+6} \mathcal{X}|_{6+8} \mathcal{X}|_{6+9} \| \mathcal{X}|_{11+13} \mathcal{X}|_{11+6} \mathcal{X}|_{11+8} \mathcal{X}|_{11+9} \\ &= \mathcal{X}|_{20} \mathcal{X}|_{13} \mathcal{X}|_{15} \mathcal{X}|_{16} \| \mathcal{X}|_{17} \mathcal{X}|_{10} \mathcal{X}|_{12} \mathcal{X}|_{13} \| \\ &\quad \mathcal{X}|_{19} \mathcal{X}|_{12} \mathcal{X}|_{14} \mathcal{X}|_{15} \| \mathcal{X}|_{24} \mathcal{X}|_{17} \mathcal{X}|_{19} \mathcal{X}|_{20} \\ &= 0010\ 0110\ 0111\ 1000 \end{aligned}$$

Given R_{Tx} and R_{Mx} for $x \in \{1, 2, 3, 4\}$, the new PadGen function is used to generate PAD_x as follows:

$$\begin{aligned} R_{Vx} &= PadGen(APwd, R_{Tx}, R_{Mx}) \\ &= a|_{w|_1} a|_{w|_2} a|_{w|_3} a|_{w|_4} \| a|_{w|_5} a|_{w|_6} a|_{w|_7} a|_{w|_8} \| a|_{w|_9} a|_{w|_{10}} a|_{w|_{11}} a|_{w|_{12}} \\ &\quad \| a|_{w|_{13}} a|_{w|_{14}} a|_{w|_{15}} a|_{w|_{16}} \\ &= d_{R_{Vx}1} d_{R_{Vx}2} d_{R_{Vx}3} d_{R_{Vx}4} \end{aligned}$$

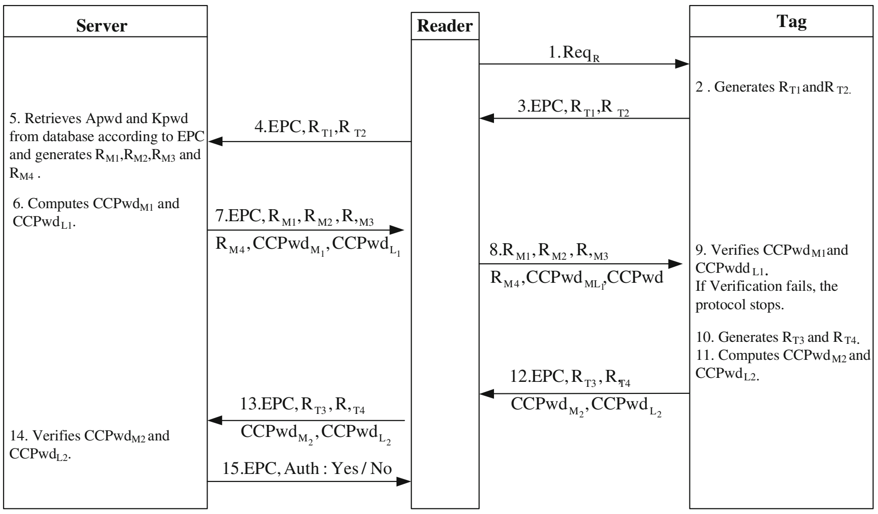

Server
Reader
Tag
1.Req_R
2 . Generates R_T1 andR_T2.
3.EPC, R_T1, R_T2
4.EPC, R_T1, R_T2
5. Retrieves Apwd and Kpwd from database according to EPC and generates R_M1, R_M2, R_M3 and R_M4 .
6. Computes CCPwd_M1 and CCPwd_L1.
7.EPC, R_M1, R_M2, R_M3 R_M4, CCPwd_M1, CCPwd_L1
8.R_M1, R_M2, R_M3 R_M4, CCPwd_ML1, CCPwd
9. Verifies CCPwd_M1 and CCPwdd_L1. If Verification fails, the protocol stops.
10. Generates R_T3 and R_T4.
11. Computes CCPwd_M2 and CCPwd_L2.
12.EPC, R_T3, R_T4 CCPwd_M2, CCPwd_L2
13.EPC, R_T3, R_T4 CCPwd_M2, CCPwd_L2
14. Verifies CCPwd_M2 and CCPwd_L2.
15.EPC, Auth : Yes / No

Fig. 1. The mutual authentication protocol proposed by HYCLT et al.

and

$$\begin{aligned} PAD_x &= PadGen(Kpwd, R_{Vx}, R_{Tx}) \\ &= k|_{z|_1}k|_{z|_2}k|_{z|_3}k|_{z|_4}||k|_{z|_5}k|_{z|_6}k|_{z|_7}k|_{z|_8}||k|_{z|_9}k|_{z|_{10}}k|_{z|_{11}}k|_{z|_{12}} \\ &\quad ||k|_{z|_{13}}k|_{z|_{14}}k|_{z|_{15}}k|_{z|_{16}} \\ &= h_{PAD_x1}h_{PAD_x2}h_{PAD_x3}h_{PAD_x4} \end{aligned}$$

where

$$\begin{aligned} w|_{1\sim4} &= d_{R_{Tx}1} + d_{R_{Mx}1}, d_{R_{Tx}1} + d_{R_{Mx}2}, d_{R_{Tx}1} + d_{R_{Mx}3}, d_{R_{Tx}1} + d_{R_{Mx}4} \\ w|_{5\sim8} &= d_{R_{Tx}2} + d_{R_{Mx}1}, d_{R_{Tx}2} + d_{R_{Mx}2}, d_{R_{Tx}2} + d_{R_{Mx}3}, d_{R_{Tx}2} + d_{R_{Mx}4} \\ w|_{9\sim12} &= d_{R_{Tx}3} + d_{R_{Mx}1}, d_{R_{Tx}3} + d_{R_{Mx}2}, d_{R_{Tx}3} + d_{R_{Mx}3}, d_{R_{Tx}3} + d_{R_{Mx}4} \\ w|_{13\sim16} &= d_{R_{Tx}4} + d_{R_{Mx}1}, d_{R_{Tx}4} + d_{R_{Mx}2}, d_{R_{Tx}4} + d_{R_{Mx}3}, d_{R_{Tx}4} + d_{R_{Mx}4} \\ z|_{1\sim4} &= d_{R_{Tx}1} + d_{R_{Vx}1}, d_{R_{Tx}1} + d_{R_{Vx}2}, d_{R_{Tx}1} + d_{R_{Vx}3}, d_{R_{Tx}1} + d_{R_{Vx}4} \\ z|_{5\sim8} &= d_{R_{Tx}2} + d_{R_{Vx}1}, d_{R_{Tx}2} + d_{R_{Vx}2}, d_{R_{Tx}2} + d_{R_{Vx}3}, d_{R_{Tx}2} + d_{R_{Vx}4} \\ z|_{9\sim12} &= d_{R_{Tx}3} + d_{R_{Vx}1}, d_{R_{Tx}3} + d_{R_{Vx}2}, d_{R_{Tx}3} + d_{R_{Vx}3}, d_{R_{Tx}3} + d_{R_{Vx}4} \\ z|_{13\sim16} &= d_{R_{Tx}4} + d_{R_{Vx}1}, d_{R_{Tx}4} + d_{R_{Vx}2}, d_{R_{Tx}4} + d_{R_{Vx}3}, d_{R_{Tx}4} + d_{R_{Vx}4} \end{aligned}$$

A more detailed description of HYCLT protocol is provided in Fig. 1 which is described as below:

1. The reader starts the protocol by sending Req_R to the tag.
2. On reception, the tag generates two random numbers R_{T1} and R_{T2} and sends its EPC with R_{T1} and R_{T2} to the reader.

3. Once the reader receipts this message, it forwards the message to the server.
4. Upon receipt the message, the server:
   – retrieves $Apwd$ and $Kpwd$ from database according to EPC;
   – generates four fresh random numbers R_{M1}, R_{M2}, R_{M3} and R_{M4};
   – computes $CCPwd_{M_1} = Apwd_M \oplus PAD_1$ and $CCPwd_{L_1} = Apwd_L \oplus PAD_2$;
   – sends $EPC, R_{M1}, R_{M2}, R_{M3}, R_{M4}, CCPwd_{M_1}$ and $CCPwd_{L_1}$ to the reader.
5. Upon receipt of the message, the reader sends R_{M1}, R_{M2}, R_{M3}, R_{M4}, $CCPwd_{M_1}$ and $CCPwd_{L_1}$ to the tag.
6. Upon receipt of the message, the tag verifies the correctness of $CCPwd_{M_1}$ and $CCPwd_{L_1}$ and does as follows:
   – generates R_{T3} and R_{T4};
   – computes $CCPwd_{M_2} = Apwd_M \oplus PAD_3$ and $CCPwd_{L_2} = Apwd_L \oplus PAD_4$;
   – and sends EPC, R_{T3}, R_{T4}, $CCPwd_{M_2}$ and $CCPwd_{L_2}$ to the reader.
7. The reader forwards the message to the server.
8. The server verifies $CCPwd_{M_2}$ and $CCPwd_{L_2}$. If they are valid it sends EPC and $Auth : Yes$ to the reader; Otherwise, it sends EPC and $Auth : No$ to the reader.

2.1 Secret Disclosure Attack on HYCLT Protocol

Considering HYCLT based on its complex PadGen function, in this section we present an attack which retrieves the secret *Access* password of any given tag in HYCLT. The presented attack is based on the following observation:

Observation 1: Assume that in Step 2 of the protocol, where the reader has started the protocol by sending Req_R to the tag and the tag generates two random numbers R_{T1} and R_{T2} and sends its EPC with R_{T1} and R_{T2} to the reader, the adversary intercepts R_{T1} and R_{T2} sent by the tag and replaces them by R'_{T1} and R'_{T2} such that, e.g., $R'_{T1} = d_{R'_{T1}1} \| d_{R'_{T1}1} \| d_{R'_{T1}1} \| d_{R'_{T1}1}$ and $R'_{T2} = d_{R'_{T2}1} \| d_{R'_{T2}1} \| d_{R'_{T2}1} \| d_{R'_{T2}1}$ where $d_{R'_{T1}1}$ or $d_{R'_{T2}1}$ could be any value $\in \{0, \ldots, 15\}$. An example is $R'_{T1} = R'_{T2} = 0$. Then we have $w|_{1\sim4} = w|_{5\sim8} = w|_{9\sim12} = w|_{13\sim16}$ and equivalently we can state that $d_{R_{Vx}1} = d_{R_{Vx}2} = d_{R_{Vx}3} = d_{R_{Vx}4}$. Consider $x = 1$, we have $d_{R_{V1}1} = d_{R_{V1}2} = d_{R_{V1}3} = d_{R_{V1}4}$ and $d_{R'_{T1}1} = d_{R'_{T1}2} = d_{R'_{T1}3} = d_{R'_{T1}4}$ On the other hand,

$$\begin{aligned} PAD_1 &= PadGen(Kpwd, R_{V1}, R'_{T1}) \\ &= k|_{z1}k|_{z2}k|_{z3}k|_{z4} \| k|_{z5}k|_{z6}k|_{z7}k|_{z8} \| k|_{z9}k|_{z10}k|_{z11}k|_{z12} \| k|_{z13}k|_{z14}k|_{z15}k|_{z16} \\ &= h_{PAD_1 1} h_{PAD_1 2} h_{PAD_1 3} h_{PAD_1 4} \end{aligned}$$

where

$$\begin{aligned} z|_{1\sim4} &= d_{R'_{T1}1} + d_{R_{V1}1}, d_{R'_{T1}1} + d_{R_{V1}2}, d_{R'_{T1}1} + d_{R_{V1}3}, d_{R'_{T1}1} + d_{R_{V1}4} \\ z|_{5\sim8} &= d_{R'_{T1}2} + d_{R_{V1}1}, d_{R'_{T1}2} + d_{R_{V1}2}, d_{R'_{T1}2} + d_{R_{V1}3}, d_{R'_{T1}2} + d_{R_{V1}4} \\ z|_{9\sim12} &= d_{R'_{T1}3} + d_{R_{V1}1}, d_{R'_{T1}3} + d_{R_{V1}2}, d_{R'_{T1}3} + d_{R_{V1}3}, d_{R'_{T1}3} + d_{R_{V1}4} \\ z|_{13\sim16} &= d_{R'_{T1}4} + d_{R_{V1}1}, d_{R'_{T1}4} + d_{R_{V1}2}, d_{R'_{T1}4} + d_{R_{V1}3}, d_{R'_{T1}4} + d_{R_{V1}4}. \end{aligned}$$

Since $d_{R'_{T1}i} + d_{R_{V1}j}$ for any i, and $j \in \{1,2,3,4\}$ is a fixed value here, we have $z|_m = z|_n$ for any m and $n \in \{1,\ldots,16\}$. Therefore we have $z|_1 = z|_2 = \ldots = z|_{16} = z$. Hence $PAD_1 = PadGen(Kpwd, R_{V1}, R'_{T1}) = k|_z \| k|_z \| \ldots \| k|_z$ where $z \in \{0,\ldots,F\}$ and $PAD_1 = PadGen(Kpwd,\ R_{V1},\ R'_{T1}) \in \{0000, FFFF\}$.

Similarly for $x = 2$ we have $d_{R_{V2}1} = d_{R_{V2}2} = d_{R_{V2}3} = d_{R_{V2}4}$ and $d_{R'_{T2}1} = d_{R'_{T2}2} = d_{R'_{T2}3} = d_{R'_{T2}4}$. On the other hand,

$$\begin{aligned} PAD_2 &= PadGen(Kpwd, R_{V2}, R'_{T2}) \\ &= k|_{z'1}k|_{z'2}k|_{z'3}k|_{z'4}\|k|_{z'5}k|_{z'6}k|_{z'7}k|_{z'8}\|k|_{z'9}k|_{z'10}k|_{z'11}k|_{z'12} \\ &\quad \|k|_{z'13}k|_{z'14}k|_{z'15}k|_{z'16} \\ &= h_{PAD_21}h_{PAD_22}h_{PAD_23}h_{PAD_24} \end{aligned}$$

where

$$\begin{aligned} z'_{1\sim4} &= d_{R'_{T2}1} + d_{R_{V2}1}, d_{R'_{T2}1} + d_{R_{V2}2}, d_{R'_{T2}1} + d_{R_{V2}3}, d_{R'_{T2}1} + d_{R_{V2}4} \\ z'_{5\sim8} &= d_{R'_{T2}2} + d_{R_{V2}1}, d_{R'_{T2}2} + d_{R_{V2}2}, d_{R'_{T2}2} + d_{R_{V2}3}, d_{R'_{T2}2} + d_{R_{V2}4} \\ z'_{9\sim12} &= d_{R'_{T2}3} + d_{R_{V2}1}, d_{R'_{T2}3} + d_{R_{V2}2}, d_{R'_{T2}3} + d_{R_{V2}3}, d_{R'_{T2}3} + d_{R_{V2}4} \\ z'_{13\sim16} &= d_{R'_{T2}4} + d_{R_{V2}1}, d_{R'_{T2}4} + d_{R_{V2}2}, d_{R'_{T2}4} + d_{R_{V2}3}, d_{R'_{T2}4} + d_{R_{V2}4}. \end{aligned}$$

Since $d_{R'_{T2}i} + d_{R_{V2}j}$ for any i and $j \in \{1,2,3,4\}$ is a fixed value here, we have $z'_m = z'_n$ for any m and $n \in \{1,\ldots,16\}$. Therefore we have $z'_1 = z'_2 = \ldots = z'_{16} = z'$. Hence $PAD_2 = PadGen(Kpwd, R_{V2}, R'_{T2}) = k|'_z \| k|'_z \| \ldots \| k|_{z'}$ where $z' \in \{0,\ldots,F\}$ and $PAD_2 = PadGen(Kpwd,\ R_{V2},\ R'_{T2}) \in \{0000, FFFF\}$.

Now given that $CCPwd_{M_1} = Apwd_M \oplus PAD_1$ and $CCPwd_{L_1} = Apwd_L \oplus PAD_2$, and there are two choices for any of PAD_1 and PAD_2 (in total 4 choices) the adversary can determine the correct $Apwd_L \| Apwd_M$ with the probability of 2^{-2}, where $Apwd = a|_0a|_1a|_2a|_3\ldots a|_{31}$, $Apwd_L = a|_0a|_1\ldots a|_{15}$ and $Apwd_M = a|_{16}a|_{17}\ldots a|_{31}$.

Following **observation 1**, we have $z|_1 = z|_2 = \ldots = z|_{16} = z$. Hence $PAD_x = PadGen(Kpwd, R_{Vx}, R'_{Tx}) = k|_z \| k|_z \| \ldots \| k|_z$ where $z \in \{0,\ldots,F\}$ and $PAD_x = PadGen\ (Kpwd, R_{Vx}, R'_{Tx}) \in \{0000, FFFF\}$. Now given that $CCPwd_{M_1} = Apwd_M \oplus PAD_1$ and $CCPwd_{L_1} = Apwd_L \oplus PAD_2$, the adversary can determine $Apwd_L \| Apwd_M$ with the probability of 2^{-2}, where $Apwd = a|_0a|_1a|_2a|_3\ldots a|_{31}$, $Apwd_M = a|_0a|_1\ldots a|_{15}$ and $Apwd_L = a|_{16}a|_{17}\ldots a|_{31}$.

3 HLL Protocol

Huang, Lin and Li in [6] have presented another EPC- C1 G2 specification complaint mutual authentication protocol with a different PadGen function and successfully verified their protocol functionality in hardware. We refer to this protocol by HLL.

In the PadGen function of the simple variant of HYCLT protocol and Konidala et al. protocol the location of a fraction of the bits of secret passwords that are included in PAD_x are decided by a public parameter which is under the adversary's control. For example, in the simple variant of HYCLT protocol, PAD_1 is calculated as $PAD_1 = PadGen(Kpwd, R_{V1}, R_{T1})$, where the location of the extraction of 8 bits out of 16 bits are determined by R_{T_1} which is under the adversary's control. The PadGen function proposed in HLL protocol is computed based on a set of values, i.e., (R_{Vx}, R_{Wx}), which is calculated inside the server or tags and they are not transmitted over the channel between the reader and the tag. There are two variants of PadGen function in HLL protocol based on XOR or MOD operation respectively. In this paper we consider the variant based on XOR and the presented attack does not work for the protocol based on MOD. To calculate PAD_1 and PAD_2 values, given two random number R_{Tx} and R_{Mx} that are generated by the tag and the server respectively, at the first an intermediate parameter denoted by $R_{Tx\oplus Mx}$ is calculated as $R_{Tx\oplus Mx} = R_{Tx} \oplus R_{Mx} = d_{R_{Tx\oplus Mx}1}d_{R_{Tx\oplus Mx}2}d_{R_{Tx\oplus Mx}3}d_{R_{Tx\oplus Mx}4}$. This parameter is used as an input for the PadGen function to calculate another temporary value denoted by R_{Wx} as follows:

$$\begin{aligned}R_{Wx} &= PadGen(APwd, R_{Tx}, R_{Tx\oplus Mx})\\ &= a|_{d_{R_{Tx}1}}a|_{d_{R_{Tx}2}}a|_{d_{R_{Tx}3}}a|_{d_{R_{Tx}4}}\|a|_{d_{R_{Tx}1}+16}a|_{d_{R_{Tx}2}+16}a|_{d_{R_{Tx}3}+16}a|_{d_{R_{Tx}4}+16}\\ &\quad \|a|_{d_{R_{Tx\oplus Mx}1}}a|_{d_{R_{Tx\oplus Mx}2}}a|_{d_{R_{Tx\oplus Mx}3}}a|_{d_{R_{Tx\oplus Mx}4}}\\ &\quad \|a|_{d_{R_{Tx\oplus Mx}1}+16}a|_{d_{R_{Tx\oplus Mx}2}+16}a|_{d_{R_{Tx\oplus Mx}3}+16}a|_{d_{R_{Tx\oplus Mx}4}+16}.\end{aligned}$$

In addition, R_{Tx} and R_{Mx} are used as the input of PadGen function to calculate a temporary value R_{Vx} as follows:

$$\begin{aligned}R_{Vx} &= PadGen(APwd, R_{Tx}, R_{Mx})\\ &= a|_{d_{R_{Tx}1}}a|_{d_{R_{Tx}2}}a|_{d_{R_{Tx}3}}a|_{d_{R_{Tx}4}}\|a|_{d_{R_{Tx}1}+16}a|_{d_{R_{Tx}2}+16}a|_{d_{R_{Tx}3}+16}a|_{d_{R_{Tx}4}+16}\|\\ &\quad a|_{d_{R_{Mx}1}}a|_{d_{R_{Mx}2}}a|_{d_{R_{Mx}3}}a|_{d_{R_{Mx}4}}\|a|_{d_{R_{Mx}1}+16}a|_{d_{R_{Mx}2}+16}a|_{d_{R_{Mx}3}+16}a|_{d_{R_{Mx}4}+16}\end{aligned}$$

Given R_{W1} and R_{V1}, PAD_1 function is calculated as follows:

$$\begin{aligned}PAD_1 &= PadGen(Kpwd, R_{V1}, R_{W1})\\ &= k|_{d_{R_{V1}1}}k|_{d_{R_{V1}2}}k|_{d_{R_{V1}3}}k|_{d_{R_{V1}4}}\|k|_{d_{R_{V1}1}+16}k|_{d_{R_{V1}2}+16}k|_{d_{R_{V1}3}+16}k|_{d_{R_{V1}4}+16}\|\\ &\quad k|_{d_{R_{W1}1}}k|_{d_{R_{W1}2}}k|_{d_{R_{W1}3}}k|_{d_{R_{W1}4}}\|k|_{d_{R_{W1}1}+16}k|_{d_{R_{W1}2}}\\ &\quad + 16k|_{d_{R_{W1}3}+16}k|_{d_{R_{W1}4}+16}\end{aligned}$$

To calculate PAD_2, the protocol at the first calculates a new parameter $R_{V1\oplus W1}$ as $R_{S1} = R_{V1\oplus W1} = R_{V1} \oplus R_{W1}$. Given R_{S1} and R_{V1}, the value of PAD_2 is calculated as follows:

$$\begin{aligned}PAD_2 &= PadGen(Kpwd, R_{V1}, R_{S1})\\ &= k|_{d_{R_{V1}1}}k|_{d_{R_{V1}2}}k|_{d_{R_{V1}3}}k|_{d_{R_{V1}4}}\|k|_{d_{R_{V1}1}+16}k|_{d_{R_{V1}2}+16}k|_{d_{R_{V1}3}+16}k|_{d_{R_{V1}4}+16}\|\\ &\quad k|_{d_{R_{S1}1}}k|_{d_{R_{S1}2}}k|_{d_{R_{S1}3}}k|_{d_{R_{S1}4}}\|k|_{d_{R_{S1}1}+16}k|_{d_{R_{S1}2}+16}k|_{d_{R_{S1}3}+16}k|_{d_{R_{S1}4}+16}\end{aligned}$$

A description of HLL protocol is provided in Fig. 2 which is described as follows:

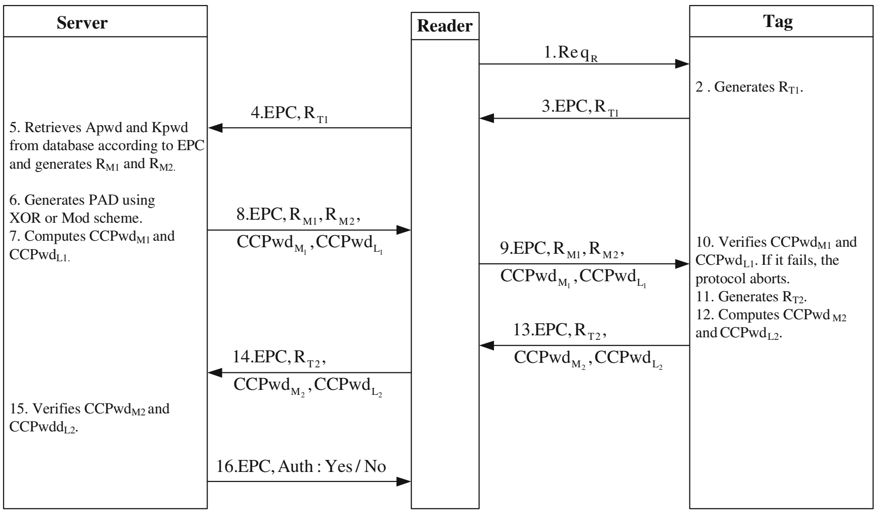

Server
Reader
Tag
1.ReqR
2 . Generates RT1.
3.EPC, RT1
4.EPC, RT1
5. Retrieves Apwd and Kpwd from database according to EPC and generates RM1 and RM2.
6. Generates PAD using XOR or Mod scheme.
7. Computes CCPwdM1 and CCPwdL1.
8.EPC, RM1, RM2, CCPwdM1, CCPwdL1
9.EPC, RM1, RM2, CCPwdM1, CCPwdL1
10. Verifies CCPwdM1 and CCPwdL1. If it fails, the protocol aborts.
11. Generates RT2.
12. Computes CCPwdM2 and CCPwdL2.
13.EPC, RT2, CCPwdM2, CCPwdL2
14.EPC, RT2, CCPwdM2, CCPwdL2
15. Verifies CCPwdM2 and CCPwddL2.
16.EPC, Auth : Yes / No

Fig. 2. The Huang-Lin-Li mutual authentication protocol using XOR or MOD scheme.

1. The reader starts the protocol by sending Req_R to the tag.
2. On reception, the tag generates a random number R_{T1} and sends its EPC with R_{T1} to the reader.
3. Once the reader receipt the message, forwards it to the server.
4. Upon receipt the message, the server :
   - retrieves $Apwd$ and $Kpwd$ from database according to EPC;
   - generates two random numbers R_{M1} and R_{M2};
   - generates PAD using XOR or MOD scheme, where we concentrate on XOR operation.
   - computes $CCPwd_{M_1} = Apwd_M \oplus PAD_1$ and $CCPwd_{L_1} = Apwd_L \oplus PAD_2$;
   - and sends EPC, R_{M1}, R_{M2}, $CCPwd_{M_1}$ and $CCPwd_{L_1}$ to the reader.
5. On receipt the message, the reader forwards the message to the tag.
6. Upon receipt the message, the tag verifies $CCPwd_{M_1}$ and $CCPwd_{L_1}$. If the equality does not exist, the protocol will stop. Otherwise it:
   - generates another random number R_{T2};
   - computes $CCPwd_{M_2} = Apwd_M \oplus PAD_3$ and $CCPwd_{L_2} = Apwd_L \oplus PAD_4$;
   - and sends $EPC, R_{T2}, CCPwd_{M_2}$ and $CCPwd_{L_2}$ to the reader.
7. The reader forwards the message to the server.
8. The server verifies $CCPwd_{M_2}$ and $CCPwd_{L_2}$. In the case of equality, sends EPC and $Auth : Yes$ to the reader. Otherwise it sends EPC and $Auth : No$ to the reader.

3.1 Security Analysis of the HLL Protocol

Passive Adversary

Observation 1: It can be seen that $d_{R_{V1}}1 = d_{R_{W1}}1 = a|_{d_{R_{T1}}1}a|_{d_{R_{T1}}2}$ $a|_{d_{R_{T1}}3}a|_{d_{R_{T1}}4}$ and $d_{R_{V1}}2 = d_{R_{W1}}2 = a|_{d_{R_{T1}}1+16}a|_{d_{R_{T1}}2+16}a|_{d_{R_{T1}}3+16}a|_{d_{R_{T1}}4+16}$.

Observation 2: Following **Observation 1**, $k|_{d_{R_{V1}}1} = k|_{d_{R_{W1}}1}$, $k|_{d_{R_{V1}}2} = k|_{d_{R_{W1}}2}$, $k|_{d_{R_{V1}}1+16} = k|_{d_{R_{W1}}1+16}$ and $k|_{d_{R_{V1}}2+16} = k|_{d_{R_{W1}}2+16}$.

Following this observation PAD_1 can be rewritten as follows:

$$\begin{aligned}PAD_1 = {}& k|_{d_{R_{V1}}1}k|_{d_{R_{V1}}2}k|_{d_{R_{V1}}3}k|_{d_{R_{V1}}4}\|k|_{d_{R_{V1}}1+16}k|_{d_{R_{V1}}2+16}k|_{d_{R_{V1}}3+16}k|_{d_{R_{V1}}4+16}\|\\ & k|_{d_{R_{V1}}1}k|_{d_{R_{V1}}2}k|_{d_{R_{W1}}3}k|_{d_{R_{W1}}4}\|k|_{d_{R_{V1}}1+16}k|_{d_{R_{V1}}2+16}k|_{d_{R_{W1}}3+16}k|_{d_{R_{W1}}4+16}.\end{aligned}$$

Given that $CCPwd_{M_1} = Apwd_M \oplus PAD_1$ and $Apwd_M = a|_{16}a|_{17}\dots a|_{31}$, we can extract the following equations:

$$\begin{aligned}(CCPwd_{M_1})|_0 \oplus (CCPwd_{M_1})|_8 &= a|_{16} \oplus a|_{24}\\ (CCPwd_{M_1})|_1 \oplus (CCPwd_{M_1})|_9 &= a|_{17} \oplus a|_{25}\\ (CCPwd_{M_1})|_4 \oplus (CCPwd_{M_1})|_{12} &= a|_{20} \oplus a|_{28}\\ (CCPwd_{M_1})|_5 \oplus (CCPwd_{M_1})|_{13} &= a|_{21} \oplus a|_{29};\end{aligned}$$

which can be used to recognize a target tag with the success probability of $1 - 2^{-4}$.

Observation 3: Following **observation 1**, one can state that $d_{R_{S1}}1 = d_{R_{S1}}2 = 0$ and $R_{V1\oplus W1} = 00d_{R_{S1}}3d_{R_{S1}}4$.

On the other hand:

$$\begin{aligned}PAD_2 = {}& k|_{d_{R_{V1}}1}k|_{d_{R_{V1}}2}k|_{d_{R_{V1}}3}k|_{d_{R_{V1}}4}\|k|_{d_{R_{V1}}1+16}k|_{d_{R_{V1}}2+16}k|_{d_{R_{V1}}3+16}k|_{d_{R_{V1}}4+16}\|\\ & k|_{d_{R_{S1}}1}k|_{d_{R_{S1}}2}k|_{d_{R_{S1}}3}k|_{d_{R_{S1}}4}\|k|_{d_{R_{S1}}1+16}k|_{d_{R_{S1}}2+16}k|_{d_{R_{S1}}3+16}k|_{d_{R_{S1}}4+16}\end{aligned}$$

Hence, we can rewrite PAD_2 as follows:

$$\begin{aligned}PAD_2 = {}& k|_{d_{R_{V1}}1}k|_{d_{R_{V1}}2}k|_{d_{R_{V1}}3}k|_{d_{R_{V1}}4}\|k|_{d_{R_{V1}}1+16}k|_{d_{R_{V1}}2+16}k|_{d_{R_{V1}}3+16}k|_{d_{R_{V1}}4+16}\|\\ & k|_0k|_0k|_{d_{R_{S1}}3}k|_{d_{R_{S1}}4}\|k|_{16}k|_{16}k|_{d_{R_{S1}}3+16}k|_{d_{R_{S1}}4+16}.\end{aligned}$$

So, $CCPwd_{L_1} = xxxx\|xxxx\|(k|_0\oplus a|_8)(k|_0\oplus a|_9)xx\|(k|_{16}\oplus a|_{12})(k|_{16}\oplus a|_{13})xx$, which can be used to recognize a target tag with the success probability of $1-2^{-4}$. This information also leaks 4 bits of secret passwords.

Observation 4: Comparing the details of PAD_1 and PAD_2 we can see that $h_{PAD_1}1 = h_{PAD_2}1$ and $h_{PAD_1}2 = h_{PAD_2}2$. Now given that $CCPwd_{M_1} = Apwd_M\oplus PAD_1$ and $CCPwd_{L_1} = Apwd_L \oplus PAD_2$, the adversary can use the 8-LSB of $CCPwd_{M_1} \oplus CCPwd_{L_1}$ as a measure to trace the target tag, which is independent of the nonces and only dependent on the $Apwd_L \oplus Apwd_M$ and is static.

More precisely (x denotes an unknown binary value):

$$\begin{aligned}
PAD_1 &= h_{PAD_1 1} h_{PAD_1 2} h_{PAD_1 3} h_{PAD_1 4} \\
PAD_2 &= h_{PAD_2 1} h_{PAD_2 2} h_{PAD_2 3} h_{PAD_2 4} \\
CCPwd_{M_1} &= Apwd_M \oplus PAD_1 \\
CCPwd_{L_1} &= Apwd_L \oplus PAD_2 \\
PAD_1 \oplus PAD_2 &= 0000\|0000\|xxxx\|xxxx; (Observation\ 4) \\
Apwd_L &= a|_0 a|_1 \ldots a|_{15} \\
Apwd_M &= a|_{16} a|_{17} \ldots a|_{31} \\
CCPwd_{L_1} \oplus CCPwd_{M_1} &= (a|_0 \oplus a|_{16})(a|_1 \oplus a|_{17})(a|_2 \oplus a|_{18})(a|_3 \oplus a|_{19})\| \\
&\quad (a|_4 \oplus a|_{20})(a|_5 \oplus a|_{21})(a|_6 \oplus a|_{22})(a|_7 \oplus a|_{23}) \\
&\quad \|xxxx\|xxxx.
\end{aligned}$$

Active Adversary. Assume that an active adversary intercepts the message from the tag to the reader in step 3 and replaces R_{T1} by $R^i_{T1} = d_{R^i_{T1}1} \|d_{R^i_{T1}2} \|d_{R^i_{T1}3} \|d_{R^i_{T1}4} = i\|i\|i\|i$, for $0 \geq i \geq 15$. Then, one can state that $d_{R^i_{V1}1} = d_{R^i_{W1}1} = a|_i a|_i a|_i a|_i \in \{0000, 1111\}$ (base 2). In addition, we assume that $k|_0 \oplus k|_{15} = k|_{16} \oplus k|_{31} = 1$. Now, given these assumptions and given $CCPwd^i_{M_1}$ and $CCPwd^j_{M_1}$, we can find out whether $a|_i = a|_j$ as follows:

$$\begin{aligned}
& CCPwd^i_{M_1} \oplus CCPwd^j_{M_1} \\
&= Apwd_M \oplus PAD^i_1 \oplus Apwd_M \oplus PAD^j_1 \\
&= PAD^i_1 \oplus PAD^j_1 \\
&= (k|_{d_{R^i_{V1}1}} \oplus k|_{d_{R^j_{V1}1}})(k|_{d_{R^i_{V1}2}} \oplus k|_{d_{R^j_{V1}2}})(k|_{d_{R^i_{V1}3}} \oplus k|_{d_{R^j_{V1}3}})(k|_{d_{R^i_{V1}4}} \oplus k|_{d_{R^j_{V1}4}})\| \\
&\quad (k|_{d_{R^i_{V1}1}+16} \oplus k|_{d_{R^j_{V1}1}+16})(k|_{d_{R^i_{V1}2}+16} \oplus k|_{d_{R^j_{V1}2}+16})(k|_{d_{R^i_{V1}3}+16} \oplus k|_{d_{R^j_{V1}3}+16}) \\
&\quad (k|_{d_{R^i_{V1}4}+16} \oplus k|_{d_{R^j_{V1}4}+16})\|(k|_{d_{R^i_{W1}1}} \oplus k|_{d_{R^j_{W1}1}})(k|_{d_{R^i_{W1}2}} \oplus k|_{d_{R^j_{W1}2}})(k|_{d_{R^i_{W1}3}} \\
&\quad \oplus k|_{d_{R^j_{W1}3}})(k|_{d_{R^i_{W1}4}} \oplus k|_{d_{R^j_{W1}4}})\|(k|_{d_{R^i_{W1}1}+16} \oplus k|_{d_{R^j_{W1}1}+16})(k|_{d_{R^i_{W1}2}+16} \\
&\quad \oplus k|_{d_{R^j_{W1}2}+16})(k|_{d_{R^i_{W1}3}+16} \oplus k|_{d_{R^j_{W1}3}+16})(k|_{d_{R^i_{W1}4}+16} \oplus k|_{d_{R^j_{W1}4}+16}).
\end{aligned}$$

Since, $k|_0 \oplus k|_{15} = 1$ and $d_{R^i_{V1}1} \in \{0, 15\}$ and $d_{R^j_{V1}1} \in \{0, 15\}$ then $(k|_{d_{R^i_{V1}1}} \oplus k|_{d_{R^j_{V1}1}}) = 0$ implies that $a|_i = a|_j$ and vice versa. Hence the adversary can fix $j = 0$ and varies i from 1 to 15 and verifies whether $a|_i = a|_0$. In this way the adversary can determine all bits of $Apwd_L$ with the success probability of $\frac{1}{2}$. Following the same approach for $(k|_{d_{R^i_{V1}2}} \oplus k|_{d_{R^j_{V1}2}})$ all bits of $Apwd_M$ can be determined with the success probability of $\frac{1}{2}$. Hence an active adversary can determine all 32 bits of $Apwd$ with success probability of 2^{-4}. On the other hand, given $Apwd$, R_{Tx} and R_{Mx} the adversary can determine PAD_1, PAD_2, R_{V1}, R_{W1} and $R_{V1 \oplus W1}$. Given this information, the adversary can also retrieve $Kpwd$.

4 Conclusion

In this paper we considered the security of two RFID mutual authentication protocols conforming to the EPC-C1 G2 standard. In these two protocols, authors aimed to solve ISO 18000-6C protocol weaknesses by using a special pad generation function named PadGen to mask tag's *Access* password $Apwd = Apwd_M \| Apwd_L$ before the data is transmitted. We showed that an attacker can obtain the *Access* and *Kill* passwords with high probability. We found that in Huang et al. scheme the adversary can determine the *Access* password with the probability of 2^{-2}, and in Huang-Lin-Li scheme the passive adversary can trace a target tag with the success probability of $1 - 2^{-4}$ and the active adversary can determine all 32 bits of *Access* password with success probability of 2^{-4}. Given this information, the adversary can also retrieve $Kpwd$. By knowing *Access* and *Kill* passwords the attacker can access the tag's memory and can make the target inoperative respectively.

Acknowledgments. We would like to thank anonymous reviewers for useful comments.

References

1. Bailey, D.V., Juels, A.: Shoehorning security into the EPC tag standard. In: De Prisco, R., Yung, M. (eds.) SCN 2006, LNCS, vol. 4116, pp. 303–320. Springer, Heidelberg (2006)
2. Chen, C.-L., Chien, C.-F.: Based on mobile RFID for membership stores system conforming EPC C1 G2 standards. IJAHUC **10**(4), 207–218 (2012)
3. Chen, C.-L., Deng, Y.-Y.: Conformation of EPC Class 1 Generation 2 standards RFID system with mutual authentication and privacy protection. Eng. Appl. AI **22**(8), 1284–1291 (2009)
4. Chien, H.-Y., Chen, C.-H.: Mutual authentication protocol for RFID conforming to EPC Class 1 Generation 2 standards. Comput. Stand. Interfaces **29**(2), 254–259 (2007)
5. EPCGlobal: Class-1 generation 2 UHF air interface protocol standard version 1.2.0, Gen2 Standard. http://www.epcglobalinc.org/standards/ (2008)
6. Huang, Y.-J., Lin, W.-C., Li, H.-L.: Efficient implementation of RFID mutual authentication protocol. IEEE Trans. Industr. Electron. **59**(12), 4784–4791 (2012)
7. Huang, Y.-J., Yuan, C.-C., Chen, M.-K., Lin, W.-C., Teng, H.-C.: Hardware implementation of RFID mutual authentication protocol. IEEE Trans. Industr. Electron. **57**(5), 1573–1582 (2010)
8. Information technology - Radio frequency identification for item management. Part 6: Parameters for air interface communications at 860 MHz to 960 MHz. http://www.iso.org/iso/catalogue_detail?csnumber=34117 (2005)
9. Konidala, D., Kim, Z., Kim, K.: A simple and cost effective RFID tag-reader mutual authentication scheme. In: Proceedings of International Conference on RFID Security, pp. 141–152, July 2007
10. Ma, C., Li, Y., Deng, R.H., Li, T.: RFID privacy: relation between two notions, minimal condition, and efficient construction. In: Al-Shaer, E., Jha, S., Keromytis, A.D. (eds.) ACM Conference on Computer and Communications, Security, pp. 54–65. ACM Press, New York (2009)

11. Park, J., Na, J., Kim, M.: A practical approach for enhancing security of EPCglobal RFID Gen2 tag. In: FGCN (1), pp. 436–441. IEEE (2007)
12. Peris-Lopez, P., Hernandez-Castro, J., Estevez-Tapiador, J., Ribagorda, A.: Practical attacks on a mutual authentication scheme under the EPC Class-1 Generation-2 standard. Comput. Commun. **32**(7–10), 1185–1193 (2009)
13. Peris-Lopez, P., Hernandez-Castro, J.C., Estevez-Tapiador, J.M., Ribagorda, A.: RFID specification revisited. In: The Internet of Things: From RFID to the Next-Generation Pervasive Networked Systems, pp. 311–346. Taylor & Francis Group, London (2008)
14. Want, R.: An introduction to RFID technology. IEEE Pervasive Comput. **5**(1), 25–33 (2006)

RFID Hardware

Dietary Recommendations for Lightweight Block Ciphers: Power, Energy and Area Analysis of Recently Developed Architectures

Lejla Batina[1,2], Amitabh Das[2], Barış Ege[1(✉)], Elif Bilge Kavun[3], Nele Mentens[2,4], Christof Paar[3], Ingrid Verbauwhede[2], and Tolga Yalçın[3]

[1] Institute for Computing and Information Sciences, Radboud University Nijmegen, Nijmegen, The Netherlands
b.ege@cs.ru.nl
[2] ESAT/COSIC, KU Leuven and iMinds, Leuven, Belgium
[3] Horst Görtz Institute for IT-Security, Ruhr University Bochum, Bochum, Germany
[4] ACRO/ES&S, Katholieke Hogeschool Limburg, Diepenbeek, Belgium

Abstract. In this paper we perform a comprehensive area, power, and energy analysis of some of the most recently-developed lightweight block ciphers and we compare them to the standard AES algorithm. We do this for several different architectures of the considered block ciphers. Our evaluation method consists of estimating the pre-layout power consumption and the derived energy using Cadence Encounter RTL Compiler and ModelSIM simulations. We show that the area is not always correlated to the power and energy consumption, which is of importance for mobile battery-fed devices. As a result, this paper can be used to make a choice of architecture when the algorithm has already been fixed; or it can help deciding which algorithm to choose based on energy and key/block length requirements.

1 Introduction

In the past decade various proposals for "lightweight" symmetric ciphers have been made. Among more carefully investigated ones are Clefia [20], HIGHT [21], KATAN [4], mCrypton [22], and PRESENT [3]. This turned into a very active area of research as evident by several algorithms proposed over the course of the past two years, including KLEIN [18], LED [1], Piccolo [2] and PRINCE [16]. The dominant metric used in the majority of the proposals has been the number of gate equivalence, or GE, needed for realizing the cipher in hardware. This number is derived by dividing the silicon area used for a cipher with a given standard-cell library by the area of a two-input NAND gate. Hence, the popular gate equivalence count can be thought of as a normalized area measure. Even though helpful, the metric does not answer all questions regarding lightweight ciphers.

The purpose of the investigation at hand is to perform a comprehensive area, power, and energy analysis of some of the most recently-developed lightweight

DOI: 10.1007/978-3-642-41332-2_7,

block ciphers along with the well-known block ciphers, which can be helpful for both the engineering and theoretical communities concerned with lightweight cryptography. Given that lightweight algorithms are particularly interesting for battery-powered or passive systems such as RFID tags, a valid energy prediction is very desirable.

In the most recent work of Kerckhoff et al. [15], the area, power consumption, throughput and energy of 6 block ciphers are evaluated. Conclusions are drawn with respect to round unrolling, parallelism and pipelining. The paper at hand covers 11 lightweight block cipher architectures (of 7 different lightweight block ciphers) and 6 different AES architectures. The differences between the AES architectures are not limited to round unrolling, parallelism and pipelining, but are based on specific design choices. This gives a more fair comparison of the standardized AES to recently-developed lightweight block ciphers.

Other related works in the past were focused mainly on other, more specific aspects of low-cost applications. For instance, the work of Singelée et al. [14] focuses on the computation and communication energy budget of authentication protocols for active RFID tags. Accordingly, they consider other cryptographic primitives that can be used for authentication i.e. ECC-based protocols. On the other hand, the AES algorithm as the main standard for encryption in the past decade, has been already evaluated on the energy consumption in several previous studies [8,11,17]. In particular the work of Tilich et al. [17] examines several different AES S-box implementations that are based on three different design strategies. The results addressed the consequences of different design strategies on critical path delay, silicon area, and power consumption.

All those works contributed to the better understanding of low-cost design principles. As mentioned above, the requirements of extreme low-cost applications of today require more lightweight solutions as advocated by the new block ciphers' proposals. Our work extends those studies with an extensive suite of recent lightweight symmetric schemes.

The paper is organized as follows. In Sect. 2, the considered block ciphers are briefly revisited together with the specific architectures. Section 3 elaborates on our analysis methodology. Finally, Sect. 4 presents and discusses the results and Sect. 5 concludes the paper.

2 Background

In this section, we provide background information on the evaluated block cipher architectures. The information is grouped according to similarities in the architectures.

2.1 Parallel Implementation of Block Ciphers

We have implemented fully parallel versions of AES-128, CLEFIA-128, PRESENT-80, LED-128, KLEIN-64, mCrypton-96 and PRINCE-128, where the

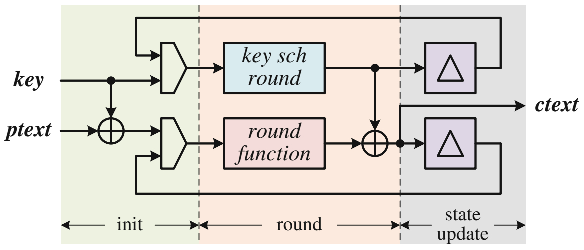


Fig. 1. Folded (round-based) implementation of a generic block cipher

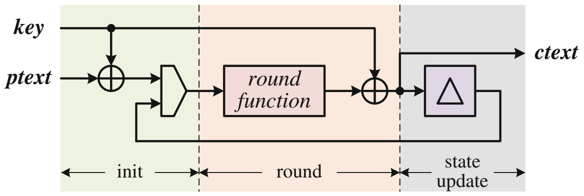


Fig. 2. Folded (round-based) implementation of a generic block cipher – no key schedule

number next to each cipher represents the key length chosen for the implementation. Among these, AES and CLEFIA are 128-bit block ciphers, whereas all others are 64-bit. For a fair comparison, we have implemented the encryption-only version of each cipher. All the implemented block ciphers share the same structure. Any such block cipher can be implemented as shown in Fig. 1, where the round function is instantiated only once. In this case, the initial input (upon a start signal) of the round function is the sum of the input key and the plaintext (i.e. the initial state). It is processed by the combinational round function and the next state is generated. It is then stored in the state register, whose output becomes the input to the round function in the next cycle. The iteration is as many rounds as the cipher is defined for. Finally, the ciphertext can be taken either from the state register output or some internal node of the round function block. In some cases, a final whitening key may also be added onto the value from the output node to generate the ciphertext (as in the case of PRINCE). The datapath width inside the round function block is equal to the cipher block size, i.e. 128 bits for AES and CLEFIA, 64 bits for all others, while the datapath width of the key scheduling block depends on the selected key size. In the case of LED, we use a fixed key (fixed means either the original input key, or a function of it). Figure 2 illustrates the no key-update case for such a round-based implementation.

This is basically the design strategy we used in all our parallel implementations. In order to keep the round numbers minimum, we got the ciphertext from internal nodes inside the round function instead of the outputs of the state registers. This way, it was ensured that the cipher could be run in maximal throughput, that is the distance between two consecutive starts is equal to the number of rounds.

We furthermore implemented parallel versions of AES in various flavors. Mainly, we focused on the S-box, which is the most area consuming unit inside the AES algorithm. The first implementation (AES_lut_128) uses lookup-table based S-boxes. The second version implements the S-box in the composite field $GF((2^4)^2)$ (AES_4_2_128), and the third version in the composite field $GF(((2^2)^2)^2)$ (AES_2_2_2_128), both as explained in [5]. We have also observed that the isomorphic transform matrices used for composite implementations are very area-consuming due to the high number of 1's inside, that correspond to XOR gates in hardware. In order to reduce these number of 1's, it is possible to use other isomorphic matrices, that are affine equivalent to the original matrices. Of course, this requires that the corresponding affine and inverse affine transformations have to be applied to the plaintext and ciphertext, respectively. Similarly, the key scheduling has to be also carried on the affine equivalent "domain". Depending on the choice of the affine transformation, it is possible to reduce the area or the power consumption or both of the overall design. We chose transforms that would minimize the power consumption, and have implemented two more flavors of AES, namely affine-transformed in the composite field $GF((2^4)^2)$ (AES_iso_4_2_128), and affine-transformed in the composite field $GF(((2^2)^2)^2)$ (AES_iso_2_2_2_128).

2.2 Unfolded Implementation of PRINCE

Since PRINCE is originally designed for unfolded implementation, i.e. all round functions are realized within a single cycle without need for a state register, we have also added an unfolded version of PRINCE. It is basically a realization of PRINCE with all 12 rounds unfolded. In our unfolded implementation strategy, the key is added to the input plaintext to generate the initial state followed by various numbers of identical rounds in order to update the state. This is shown in Fig. 3 for the unfolded version of a generic three-round block cipher. However, as PRINCE uses a fixed key, we actually implement the no key-update version of this implementation. Figure 4 illustrates this case.

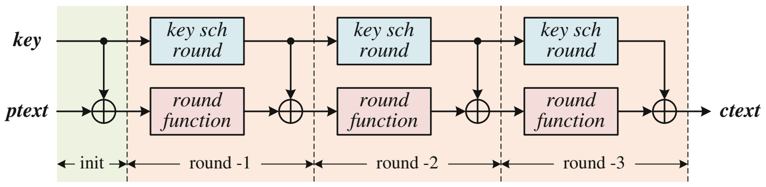


Fig. 3. Unfolded implementation of a generic 3-round block cipher

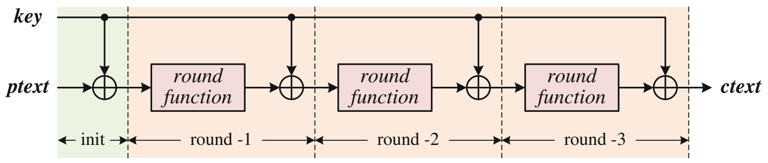


Fig. 4. Unfolded implementation of a generic 3-round block cipher – no key schedule

2.3 Implementation of KATAN

KATAN is a block cipher that belongs to a family of small and efficient hardware-oriented block ciphers. KATAN ciphers include KATAN32, KATAN48, and KATAN64. All three ciphers use 80-bit keys and have a different block size (KATANn has an n-bit block size). All three block ciphers are highly compact,with the one having the smallest block size resulting in the smallest circuit area of only 801 GEs.

In KATAN, the plaintext is loaded in two registers. In each round, several bits are taken from the registers and entered in two nonlinear Boolean functions. The output of the Boolean functions is loaded to the least significant bits of the registers (after they are shifted). This is done in an invertible manner. To ensure sufficient mixing, 254 rounds of the cipher are executed. A round counting LFSR is used instead of a counter, for counting the rounds to stop the encryption after 254 rounds, and to introduce more diffusion as well. As there are 254 rounds, an 8-bit LFSR with a sparse feedback polynomial can be used. The LFSR is initialized with some state, and the cipher has to stop running the moment the LFSR arrives to the predetermined state. The key schedule of the KATAN cipher loads the 80-bit key into another LFSR (the least significant bit of the key is loaded to position 0 of the LFSR). In each round, positions 79 and 78 of the LFSR are generated as the round's subkey, and the LFSR is clocked twice [4].

2.4 Compact Implementation of AES

The compact AES core, as described by Moradi et al. [12] (AES_small_core), is byte-based. It uses only one S-box, which is implemented using composite field arithmetic in $GF(((2^2)^2)^2)$ [5]. Note that Moradi et al. suggest to use scan-flip-flops, while we follow the conventional tool flow that does not introduce a scan-flip-flop for the combination of a multiplexer and a flip-flop. We also did not make an effort in reducing the area of the control logic, which results in an area (reported in Table 1) that is larger than the area reported by Moradi et al.

3 Analysis Strategy

3.1 Architectural Decisions

In order to perform a fair comparison, we make the following architectural decisions:

- All inputs and outputs of the cipher are buffered through a flip-flop.
- We consider encryption-only architectures. This is justified by the fact that the most popular modes of operation do not need decryption [24].

3.2 Evaluation of Design Parameters

The area reports are generated after hierarchical synthesis in *Cadence Encounter RTL Compiler* using UMC 130 nm low-leakage Faraday technology library. The area numbers in the tables are given in terms of two-input NAND gate equivalents (GEs).

Each module is first synthesized for the best power. We used *Cadence Encounter RTL Compiler* in this step. The implementations have been synthesized in UMC 130 nm low-leakage Faraday technology library. The generated netlists are then used to simulate the actual module with 100 random keys together with 10 random plaintexts per key to get the best statistics. From these simulations, SAIF files are generated, which also contain the toggle counts. All simulations are performed using Modelsim. In the last step, the SAIF files are sent back to the synthesis tool together with the netlist from the initial synthesis to run power analysis.

4 Results

In this section, we list the generated design properties for the different architectures. Further, we detect anomalies based on the fact that we expect the dynamic power consumption to be larger for designs with a larger area. The anomalies represent architectures of which (some) gates contribute less to the static and/or dynamic power consumption than the gates of other architectures. This is mainly related to the number of transistors that are conducting (for the static power consumption) and the number of nodes that switch (for the dynamic power consumption). The energy per bit is calculated by dividing the total power by the clock frequency and then multiplying by the cycle count, followed by dividing by the block length.

Table 1. Performance and energy numbers of various AES realizations all using 128-bit block lengths.

Cipher architecture	Block length	Encryption time (# cycles)	Frequency (KHz)	Area (GEs)	Static power (μW)	Dynamic power (μW)	Energy per bit (pJ/bit)	Energy (nJ)
AES_small_core	128	211	100	3685	6.25	11.31	289.47	37.05
AES_2_2_2_128	128	10	100	12405	24.46	210.15	183.29	23.46
AES_4_2_128	128	10	100	11453	21.37	135.26	122.37	15.66
AES_iso_2_2_2_128	128	10	100	15442	30.41	52.85	65.05	8.33
AES_iso_4_2_128	128	10	100	13052	25.19	37.06	48.63	6.23
AES_lut_128	128	10	100	19591	30.81	96.11	99.16	12.69

Table 2. Performance and energy numbers of lightweight block ciphers implementations.

Cipher architecture	Block/Key length	Encryption time (# cycles)	Frequency (KHz)	Area (GEs)	Static power (μW)	Dynamic power (μW)	Energy per bit (pJ/bit)	Energy (nJ)
CLEFIA	128/128	18	100	6941	13.24	37.09	70.78	9.06
KLEIN_parallel	64/64	12	100	2760	4.88	2.18	13.24	0.85
KLEIN_serial	64/64	98	100	1432	2.56	1.48	61.86	3.96
LED	64/128	48	100	3194	5.62	2.34	59.70	3.82
mCrypton	64/96	13	100	3197	5.80	2.50	16.86	1.08
PRESENT	64/80	31	100	2195	3.75	1.14	23.69	3.82
PRINCE_folded	64/128	12	100	2953	5.75	2.80	16.03	1.03
PRINCE_unfolded	64/128	1	100	39938	16.13	120.20	21.30	1.36
Katan_32	32/80	254	100	801	1.52	0.43	154.78	4.94
Katan_48	48/80	254	100	925	1.71	0.49	116.42	5.60
Katan_64	64/80	254	100	1048	1.94	0.56	99.22	6.34

A comparison of the AES architectures in Table 1 shows that AES_small_core consumes less area and power than the other 5 parallel cores, as expected. However, because of the large number of required cycles to perform the computation, the energy consumption is higher than the parallel architectures. The parallel architectures only differ in the way the S-box was implemented. The table shows that the architectural differences in the S-boxes cause significant differences in area, power and energy. The reason for some architectures to have a larger dynamic power consumption compared to others while they have a smaller area is probably caused by the fact that these architectures give rise to more internal glitches. The reason for some architectures to have a larger static power consumption compared to others while they have a smaller area is probably caused by the fact that parts of the architecture are not being used all the time during the computation.

KLEIN_parallel in Table 2 is a round-wise implementation which processes 1 round of KLEIN-64 in one clock cycle. However, KLEIN_serial in the same table is a byte serialized implementation of the same version of KLEIN. Since some of the resources are re-used in the serial implementation, the dynamic power is decreased by half. However, this has a negative effect on the number of rounds that need to be run and therefore we can see the serialized approach is not energy efficient. In the KATAN designs, the block size changes while the key size is the same. Therefore a slight change in area and also in static power consumption is observed in Table 2. But the dynamic power is quite low due to the simplistic round operations included in KATAN. When comparing LED and CLEFIA, both with block length 128, it is noticeable that the area ratio is much smaller than the power consumption ratio, in favor of LED. The reason is that LED is based on a fixed key and a simpler round computation compared to CLEFIA. The energy comparison also turns out in favor of LED. In the same way, the round function of PRESENT is much simpler in terms of logic depth compared to mCrypton, resulting in a similar trend. The energy comparison turns out in favor of mCrypton though. This is due to the fact that PRESENT needs more cycles. The reason that the unfolded version of PRINCE is only 13 times larger than the folded version but consumes 60 times more power is because of the fact that the unfolded version does not contain any registers. A comparison of the energy consumption also turns out in favor of the folded version.

Note that all the implementations of the ciphers and architectures listed in Tables 1 and 2 are re-implemented from the references.

5 Conclusions and Future Work

In this paper, we evaluated the area, power consumption, and energy of 11 lightweight block cipher architectures (parallel, serial, and unfolded implementations) and 6 AES architectures (1 byte-based core and 5 parallel cores). We discussed the differences in dynamic power consumption in relation to the area. The results show that the parallel AES core with LUT-based S-boxes is the largest in terms of GEs, but consumes the least dynamic power of all parallel cores. For the other ciphers, the comparison between parallel and serialized versions depends on the algorithm, namely on the complexity of the round function.

It is evident from the results presented in this paper, dynamic power consumption plays an important role on the energy/power consumption of cryptographic chips. Although in this work, industrial tools are used to generate dynamic power consumption results, one can also investigate the HDL implementation of a cipher and estimate the dynamic power consumption through measuring the toggle activity. The basic idea is to measure the total number of bit toggles, i.e., bit flips, that happen during the encryption of a single block of a given algorithm. This metric can be in particular helpful when considering energy. The energy consumption in CMOS circuits is dominated by the dynamic power dissipation. This is directly proportional to the number of bit toggles and hence provides a good prediction for the energy consumption of a cipher. As future work, we will investigate the relation between the number of toggles from an HDL implementation and power reports from synthesis. We believe coming up with a high-level method to estimate power consumption of a cryptographic chip is very important and we will argue this in our future work as well.

Acknowledgments. This work was supported in part by Technology Foundation STW, The Netherlands as STW project SIDES and by the Research Council KU Leuven: TENSE (GOA/11/007), by iMinds, by the Flemish Government, FWO G.0550.12N and by the Hercules Foundation AKUL/11/19.

References

1. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The LED block cipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 326–341. Springer, Heidelberg (2011)

2. Shibutani, K., Isobe, T., Hiwatari, H., Mitsuda, A., Akishita, T., Shirai, T.: *Piccolo*: an ultra-lightweight blockcipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 342–357. Springer, Heidelberg (2011)
3. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
4. De Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN — a family of small and efficient hardware-oriented block ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272–288. Springer, Heidelberg (2009)
5. Canright, D.: A very compact S-Box for AES. In: Rao, J.R., Sunar, B. (eds.) CHES 2005. LNCS, vol. 3659, pp. 441–455. Springer, Heidelberg (2005)
6. Feldhofer, M., Wolkerstorfer, J., Rijmen, V.: AES implementation on a Grain of sand. IEE Proc. Inf. Secur. **152**(1), 13–20 (2005)
7. Hein, D., Wolkerstorfer, J., Felber, N.: ECC is ready for RFID – a proof in silicon. In: Avanzi, R., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 401–413. Springer, Heidelberg (2009)
8. Hodjat, A., Verbauwhede, I.: The energy cost of embedded security for wireless sensor networks. In: Griffin, G., La Porta, T., Phoha, S. (eds.) Sensor Network Operations, pp. 510–522. Wiley, New York (2006)
9. Knezevic, M.: Efficient hardware implementations of cryptographic primitives. Ph.D. thesis, Katholieke Universiteit Leuven, Belgium, 208 pp (2011)
10. Lee, Y.K., Sakiyama, K., Batina, L., Verbauwhede, I.: Elliptic curve based security processor for RFID. IEEE Trans. Comput. **57**(11), 1514–1527 (2008)
11. de Meulenaer, G., Gosset, F., Standaert, F.-X., Pereira, O.: On the energy cost of communications and cryptography in wireless sensor networks (extended version). In: IEEE International Workshop on Security and Privacy in Wireless and Mobile Computing, Networking and Communications (SecPriWiMob '08), October 2008, pp. 580–585. IEEE, New York (2008)
12. Moradi, A., Poschmann, A., Ling, S., Paar, C., Wang, H.: Pushing the limits: a very compact and a threshold implementation of AES. In: Patterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 69–88. Springer, Heidelberg (2011)
13. Rolfes, C., Poschmann, A., Leander, G., Paar, C.: Ultra-lightweight implementations for smart devices – security for 1000 gate equivalents. In: Grimaud, G., Standaert, F.-X. (eds.) CARDIS 2008. LNCS, vol. 5189, pp. 89–103. Springer, Heidelberg (2008)
14. Singelée, D., Seys, S., Batina, L., Verbauwhede, I.: The communication and computation cost of wireless security - extended abstract. In: Tsudik, G., Asokan, N. (eds.) Proceedings of the 4th ACM Conference on Wireless Network Security (WiSec '11), pp. 1–3. ACM, New York (2011)
15. Kerckhof, S., Durvaux, F., Hocquet, C., Bol, D., Standaert, F.-X.: Towards green cryptography: a comparison of lightweight ciphers from the energy viewpoint. In: Prouff, E., Schaumont, P. (eds.) CHES 2012. LNCS, vol. 7428, pp. 390–407. Springer, Heidelberg (2012)
16. Borghoff, J., et al.: PRINCE – a low-latency block cipher for pervasive computing applications. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 208–225. Springer, Heidelberg (2012)
17. Tillich, S., Feldhofer, M., Popp, T., Großschädl, J.: Area, delay, and power characteristics of standard-cell implementations of the AES S-box. Sig. Process. Syst. **50**(2), 251–261 (2008)

18. Gong, Z., Nikova, S., Law, Y.W.: KLEIN: a new family of lightweight block ciphers. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 1–18. Springer, Heidelberg (2012)
19. Hämäläinen, P., Alho, T., Hännikäinen, M., Hämäläinen, T.D.: Design and implementation of low-area and low-power AES encryption hardware core. In: Proceedings of the 9th EUROMICRO Conference on Digital System Design (DSD'06), pp. 577–583 (2006)
20. Shirai, T., Shibutani, K., Akishita, T., Moriai, S., Iwata, T.: The 128-Bit blockcipher CLEFIA (Extended Abstract). In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 181–195. Springer, Heidelberg (2007)
21. Hong, D., et al.: HIGHT: a new block cipher suitable for low-resource device. In: Goubin, L., Matsui, M. (eds.) CHES 2006. LNCS, vol. 4249, pp. 46–59. Springer, Heidelberg (2006)
22. Lim, C.H., Korkishko, T.: mCrypton – a lightweight block cipher for security of low-cost RFID tags and sensors. In: Song, J., Kwon, T., Yung, M. (eds.) WISA 2005. LNCS, vol. 3786, pp. 243–258. Springer, Heidelberg (2006)
23. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The LED block cipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 326–341. Springer, Heidelberg (2011)
24. Dworkin, M.: NIST Recommendation for Block Cipher Modes of Operation, Methods and Techniques. NIST Special Publication 800-38A (2001)

An Improved Hardware Implementation of the Quark Hash Function

Shohreh Sharif Mansouri(✉) and Elena Dubrova

Department of Electronic Systems, Royal Institute of Technology, Stockholm, Sweden
{shsm, dubrova}@kth.se

Abstract. We present an implementation of U-Quark, the lightest instance of the Quark family of hash functions, which is optimized for throughput. The throughput is increased by converting the Feedback Shift Registers (FSRs) of Quarks permutation block from the original Fibonacci configuration to the Galois configuration. In this way, the complex feedback functions of the FSRs are decomposed into several simpler feedback functions. As a result, the throughput of U-Quark is increased by 34 % on average without any area penalty. The power consumption of the hash function also improves by 19 %.

1 Introduction

The Quark family of cryptographic hash functions [1] is based on a *sponge construction*, an architecture that minimizes memory requirements and targets the implementation of cryptographic algorithms in highly-constrained environments such as RFID systems [2]. As a sponge construction, Quark can be used for message authentication, stream encryption or authenticated encryption.

The permutation block of Quark is based on shift registers and is inspired by two low-weight ciphers: the stream cipher Grain [3–5] and the block cipher KATAN [6].

Three Quark instances have been proposed [1]: U-Quark, which is the smallest design and provides at least 64-bit security against crypto-attacks such as collisions, multicollisions, distinguishers, etc.; D-Quark, which is the second lightest instance of Quark and provides at least 80-bit security against the same attacks; S-Quark, which is the heaviest instance and provides at least 112-bits security against the attacks. For all three instances, the level of security against pre-image attacks is double compared to the other attacks (i.e. it is 128 bits for U-Quark, 160 bits for D-Quark and 224 bits for S-Quark).

In this work we optimize the implementation of Quark in terms of throughput. To do so, we modify Quark's permutation block without changing its functionality (and thus also without losing any of its security properties). We transform the Non-Linear Feedback Shift Registers (NLFSRs) of the permutation block, originally given in Fibonacci configuration, into Galois NLFSRs. Fibonacci NLFSRs have feedback only on the input state bit while Galois NLFSRs have

M. Hutter and J.-M. Schmidt (Eds.): RFIDsec 2013, LNCS 8262, pp. 113–127, 2013.
DOI: 10.1007/978-3-642-41332-2_8,

several simpler feedbacks on different state bits: the latter are therefore normally characterized by better throughput. A Fibonacci-to-Galois transformation for NLFSRs was described in [7]. Here we extend this original transformation so that it allows dealing with external inputs, multiple outputs and parallel loading, all features that are needed for U-Quark. Due to the characteristics of U-Quark, we also need to modify the structure of the hash function for the transformation to be successful. This modification does not modify the functionality of the hash function but introduces a 9 cycles latency for the generation of the hash in case the initial state of the hash function is not fixed and can change from run to run.

We limit our analysis and our experiments to U-Quark. However, since all three Quark instances have the same hardware structure, the presented technique can be applied also to D-Quark and S-Quark.

The hash function throughput is increased by 34 %, without any area overhead, while its power consumption is also reduced by 19 %.

2 The Quark Hash Function

2.1 General Structure

The operation of a sponge construction is shown in Fig. 1.

A sponge construction is characterized by different parameters: the output length n, the rate (or block size) r and the capacity c [1,2].

The size of the internal state of a sponge construction is given by the width $b = r + c$. The state bits are denoted as:

$$s_0, s_1, ..., s_{b-1}$$

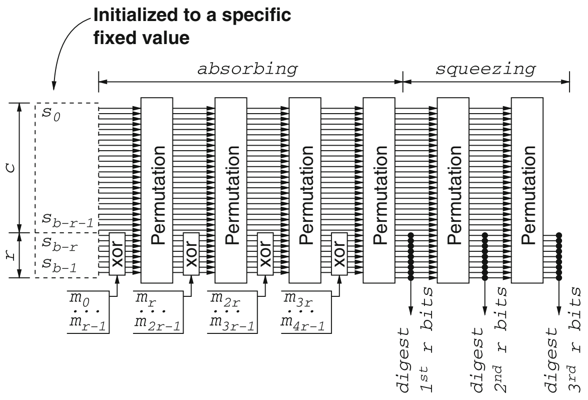


Fig. 1. Operation of a sponge construction

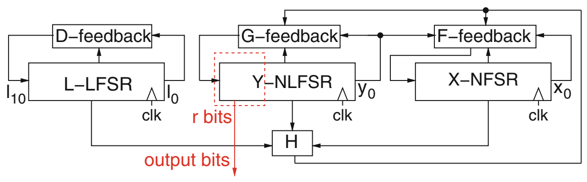


Fig. 2. Quark's permutation block

A sponge construction goes through three phases:

- *Initialization*: The initial state of the hash function is set to a fixed value, which is specific for every Quark instance; the message is padded by appending a '1' bit followed by the minimal number of '0' bits, so that the length of the message is a multiple of r.
- *Absorbing phase*: One r-bit message block is XORed with the last r bits of the state $(s_{b-r}, ..., s_{b-1})$. Then, a permutation P is applied to all the state bits. The process is repeated until all message bits have all been "absorbed".
- *Squeezing phase*: the last r bits $s_{b-r}, ..., s_{b-1}$ of the state are returned as output, then the permutation P is applied, and the process is repeated until n bits have been "squeezed out".

All instances of Quark are parametrized by different values of n, r and c. For all instances of Quark, n is constrained to be equal to $b = c + r$.

The sponge construction can be implemented serially, with a single permutation block. In alternative, to increase throughput, it can be parallelized by introducing more than one permutation block. This last solution is not taken into account in this work.

2.2 Permutation

Quark's permutation block is inspired by Grain [3–5] and KATAN [6]. Its structure is shown in Fig. 2.

The permutation block contains two $b/2 = m$ NLFSRs, called X-NLFSR and Y-NLFSR, whose state bits are respectively denoted as:

$$x_0, x_1, ..., x_{m-1}$$

and

$$y_0, y_1, ..., y_{m-1}$$

It also contains a $k = \lceil log_2(4b) \rceil$ LFSR, called L-LFSR, whose state bits are denoted as:

$$l_0, l_1, ..., l_{k-1}$$

Thus, the permutation block contains $b + k$ state bits split between the three feedback shift registers. These should not be confused with the b state bits $s_0, s_1, ..., s_{b-1}$ of Quark described in Sect. 2.1.

A permutation P takes $4b$ cycles to complete. The permutation is split between three phases (not to be confused with the sponge construction phases described in Sect. 2.1):

- *Initialization*: The X-NLFSR is initialized with the $b/2$ lowest-grade state bits of Quark $(x_0 = s_0, ..., x_{m-1} = s_{m-1})$. The Y-NLFSR is initialized with the $b/2$ second lowest-grade state bits $(y_0 = s_m, ..., y_{m-1} = s_{2m-1})$. The L-LFSR is initialized with all ones $(l_0 = 1, ..., l_{k-1} = 1)$.
- *State Update*: The permutation block is clocked for $4b$ cycles. For the X-NLFSR, elements x_i are updated to x_{i+1} for $0 \leq i < m-1$; x_{m-1} is updated to $f(x, y) \oplus h(x, y, l)$. For the Y-NLFSR, elements y_i are updated to y_{i+1} for $0 \leq i < m-1$; y_{m-1} is updated to $g(y) \oplus h(x, y, l)$. For the L-LFSR, l_i is updated to l_{i+1} for $0 \leq i < k-1$; l_{k-1} is updated to $d(l)$. The functions $f(x, y)$, $g(y)$, $d(l)$ and $h(x, y, l)$ vary between the three Quark instances. U-Quark's functions are reported in Sect. 2.3.
- *Output Write-Back*: The updated state bits of Quark are read from the X-NLFSR and the Y-NLFSR. The $b/2$ lowest-grade state bits are read from the X-NLFSR $(s_0 = x_0, ..., s_{m-1} = x_{m-1})$. The $b/2$ second lowest-grade state bits are read from the Y-NLFSR $(s_m = y_0, ..., s_{2m-1} = y_{m-1})$.

2.3 U-Quark

For U-Quark, $r = 8, c = 128$ and $n = b = 136$. The LFSR contains $k = \lceil log_2(4b) \rceil = 10$ state bits. The functions $f(x, y)$, $g(y)$, $d(l)$ and $h(x, y, l)$ are respectively equal to:

$$g(y) = y_0 \oplus y_7 \oplus y_{16} \oplus y_{20} \oplus y_{30} \oplus y_{35} \oplus y_{37} \oplus y_{42} \oplus y_{51} \oplus \\ y_{54} \oplus y_{49} \oplus y_{58}y_{54} \oplus y_{37}y_{35} \oplus y_{15}y_7 \oplus y_{54}y_{51}y_{42} \oplus \\ y_{35}y_{30}y_{20} \oplus y_{58}y_{42}y_{30}y_7 \oplus y_{54}y_{51}y_{37}y_{35} \oplus y_{58}y_{54}y_{20}y_{15} \oplus \\ y_{58}y_{54}y_{51}y_{42}y_{37} \oplus y_{35}y_{30}y_{20}y_{15}y_7 \oplus y_{51}y_{42}y_{37}y_{35}y_{30}y_{20}$$

$$f(x, y) = y_0 \oplus x_0 \oplus x_9 \oplus x_{14} \oplus x_{21} \oplus x_{28} \oplus x_{33} \oplus x_{37} \oplus \\ x_{45} \oplus x_{52} \oplus x_{55} \oplus x_{50} \oplus x_{59}x_{55} \oplus x_{37}x_{33} \oplus x_{15}x_9 \oplus \\ x_{55}x_{52}x_{45} \oplus x_{33}x_{28}x_{21} \oplus x_{59}x_{45}x_{28}x_9 \oplus x_{55}x_{52}x_{37}x_{33} \oplus \\ x_{59}x_{55}x_{21}x_{15} \oplus x_{59}x_{55}x_{52}x_{45}x_{37} \oplus x_{33}x_{28}x_{21}x_{15}x_9 \oplus \\ x_{52}x_{45}x_{37}x_{33}x_{28}x_{21}$$

$$h(x, y, l) = l_0 \oplus x_1 \oplus y_2 \oplus x_4 \oplus y_{10} \oplus x_{25} \oplus x_{31} \oplus y_{43} \oplus x_{56} \oplus \\ y_{59} \oplus y_3x_{55} \oplus x_{46}x_{55} \oplus x_{55}y_{59} \oplus y_3x_{25}x_{46} \oplus y_3x_{46}x_{55} \oplus \\ y_3x_{46}y_{59} \oplus l_0x_{25}x_{46}y_{59} \oplus l_0x_{25}$$

$$d(l) = l_0 \oplus l_3$$

3 Intuitive Idea

To improve the throughput of Quark's hash function, we need to identify the location of the critical path in the synthesized design, i.e. the longest combinational propagation delay which determines the throughput of the system.

The longest combinational delays in U-Quark are all located within the NLFSRs in the permutation block, i.e. they are all paths starting from a flip-flop in the Y-NLFSR or the X-NLFSR, passing through the $f(x, y)$, the $g(y)$ and the $h(x, y, l)$ functions and ending on a flip-flop of the X-NLFSR or the Y-NLFSR. If these paths can be made faster, Quark's performances will improve.

To speed up the paths, in Sect. 5 we transform the Fibonacci NLFSRs of the hash function into Galois NLFSRs, while at the same time transforming the $h(x, y, l)$ function so that it has lower propagation delays compared to the original function. We use an extended version of the Fibonacci-to-Galois FSR transformation proposed in [8], which also supports external input signals, multiple outputs and efficient parallel loading. We also modify the structure of Quark (without any functional modification) due to the impossibility of retrieving the highest-grade bits of the Y register in the Galois hash function. This transformation adds a 9 cycles overhead in the number of cycles required to calculate the hash in case the initial state is not fixed and can change from run to run, but does not modify the functionality of the system and thus does not have any effect on its security properties.

4 NLFSR Preliminaries

In this paper we define as Feedback Shift Register (FSR) a register composed of a sequence of n state bits x_i (note that the notation used in this section has no relation with the notation used in Sect. 2) that has certain properties. The next value f_i of every state bit in the FSR is calculated as a function of the current FSR state (the sequence of all x_i) and, possibly, some external inputs y_i. Functions f_i are said to be *update functions.*

4.1 Fibonacci and Galois FSRs

FSRs can be categorized into Fibonacci FSRs and Galois FSRs.

In a Fibonacci FSR, all update functions have the form $f_i = x_{i-1}$ except the first, for which $f_{n-1} = g(x, y)$:

$$\begin{aligned} &f_{n-1} = g(x, y) \\ &f_{n-2} = x_{n-1} \\ &\ldots \\ &f_1 = x_2 \\ &f_0 = x_1 \end{aligned}$$

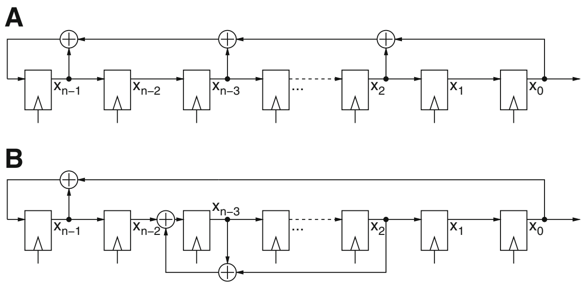


Fig. 3. (A) A Fibonacci FSR; (B) A Galois FSR

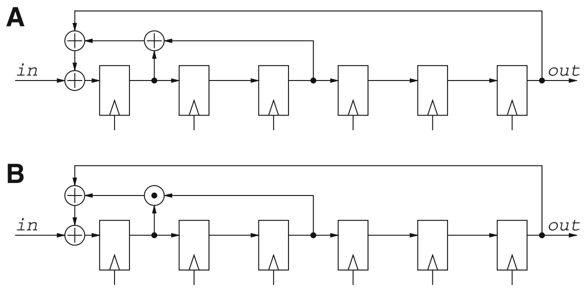


Fig. 4. (A) LFSR; (B) NLFSR

In a Galois FSR the update functions are in the form:

$$
\begin{aligned}
&f_{n-1} = g_{n-1}(x, y)\\
&f_{n-2} = x_{n-1} + g_{n-2}(x, y)\\
&...\\
&f_1 = x_1\\
&f_0 = x_0
\end{aligned}
$$

Example Fibonacci and Galois FSRs are shown in Fig. 3. Fibonacci FSRs are thus special cases of Galois FSRs.

4.2 LFSRs and NLFSRs

FSRs can be categorized into Linear FSRs (LFSRs) and Non-Linear FSRs (NLFSRs) [9].

In a linear FSR, all update functions are sums (XORs) of the state bits and the external input bits.

In a Non-Linear Feedback Shift Registers (NLFSRs), the update functions are sums of product terms, with the product terms being products (ANDs) of state bits and external input bits.

Example LFSRs and NLFSRs are shown in Fig. 4.

4.3 Equivalent FSRs

Two FSRs are said to be equivalent if the sequence of all their *output bits* are always identical [7]. Often the only output bit of an FSR is bit x_0, i.e. the last bit in the FSR chain. However, when FSRs are used in cryptographic systems such as stream ciphers or hash functions, often many of their state bits are used as output bits, i.e. inputs for external functions.

5 NLFSRs Transformation

A Galois FSR has usually better timing compared to an equivalent Fibonacci FSR, due to the fact that it has several simple update functions instead of a single, complex one [7,8,10].

There exist standard and well-known techniques to transform a Fibonacci LFSR into an equivalent Galois LFSR [7]. A transformation for NLFSRs has been proposed in [7].

5.1 Transformation Overview

The transformation in [7] considers only FSRs in which the output corresponds to the last state bit x_0 only.

The transformation in [7] consists in "moving" a set of product terms P from any update function f_i to any update function f_j with $j < i$. The transformation is restricted to product terms containing only variables x_i and no external inputs y_i. When moving a product term, the indexes of each variable x_k in each product term in P is changed to x_{k-i+j}. The transformation can be applied multiple times, i.e. it is possible to move some product terms from update function f_i to update function f_j and then move some other products from update function f_i to update function f_k.

To guarantee the equivalence of a Fibonacci NLFSR to a Galois NLFSR, it is sufficient that no product term is shifted to an update function of grade lower than the *minimum terminal bit* τ_{min}, which is calculated as:

$$\tau_{min} = \max_{p_i \in P_T} \left(max\_index\,(p_i) - min\_index\,(p_i) \right)$$

where P_T is the set of all product terms; $min\_index(p_i)$ and $max\_index(p_i)$ denote respectively the minimum and maximum index of the variables x_k in product term p_i.

Note that an "implicit" constraint when moving the products is that no product term p_i can be moved to an update function of grade lower than $n - 1 - min\_index(p_i)$ (which would result into at least one variable having a negative index).

Proof of equivalence between the Fibonacci and the Galois FSRs can be found in [7].

Figure 5 shows a Fibonacci FSR and an equivalent Galois FSR in which one product term has been moved.

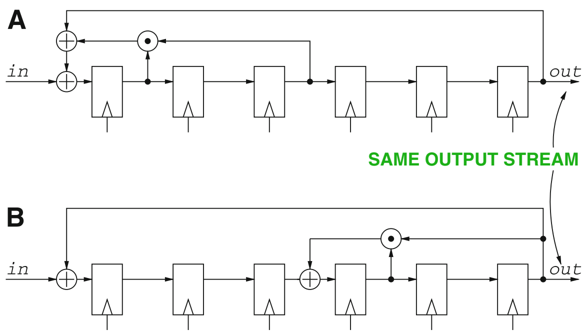


Fig. 5. (A) Original Fibonacci NLFSR; (B) Galois NLFSR

5.2 Multiple Outputs

If an FSR has multiple outputs, i.e. some of its internal state bits except the last are used as inputs for some external functions, then the transformation proposed in [7] cannot guarantee the equivalence between the original and the transformed FSRs.

However, we note that the equivalence can be guaranteed as long as no product term is moved to an update function having grade higher than Mo, where Mo denotes the grade of the highest-grade state bit x_{Mo} used as an input for an external block. In other words, the lowest-grade non-trivial update function must have a higher grade than that of the highest-grade state bit which is an output of the FSR.

Figure 6 shows a correct and an incorrect Fibonacci-to-Galois transformation.

5.3 Moving External Inputs

If an FSR receives as input some external bits y_i which are part of another FSR composed of a cascade of flip-flops, then we note that it is possible to move product terms containing a combination of x and y bits from update function i to update function j if, at the same time as the indexes of the x terms are modified, the indexes of the y_i bits are also decreased from y_i to y_{i-j}.

Figure 7 shows how a product term containing an input coming from an external FSR can be moved.

5.4 Parallel Loading

For the original and the transformed FSRs to be equivalent (having the same output stream), care must be taken to how the FSRs are loaded. If the FSRs are loaded serially, then no modification is needed. If the FSRs are loaded in parallel, then it is necessary to modify the initial value that is loaded in the Galois FSR (the initial values of the Fibonacci and Galois FSRs must be different for the output streams to be identical). This solution was described in [11]. The

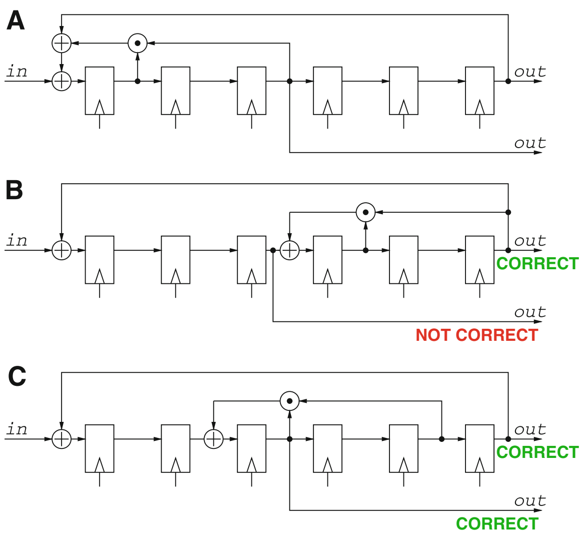


Fig. 6. (A) Original Fibonacci FSR; (B) Incorrectly transformed FSR (a non-trivial update function is righter than where the first output bit is taken); (C) Correctly transformed FSR (the last non-trivial output function is lefter than where the first output bit is taken)

computation is well-suited for fixed initialization values but is too costly in terms of hardware to do on-the-fly for values that are unknown before run time.

We here note that it is possible to modify the structure of the Galois FSR so that it can be loaded in parallel with the same value of the Fibonacci FSR, while preserving the equivalence between the FSRs. The Fibonacci FSR and the Galois FSR are loaded in parallel with the same value, but the update functions of the Galois FSR are "turned on" one by one: in the first cycle, all update functions except the highest-graded are forced to be trivial, in the second all update function except the first two are forced to be trivial, and so on until all update functions are turned on. The outputs of the two FSRs are then identical.

An example Fibonacci FSR and an equivalent parallel-loading Galois FSR are shown in Fig. 8. The enable signals for the update functions are indicated in red. The two FSRs are loaded with the same initial value. The update functions of the Galois FSR are turned on one by one. The outputs of the FSRs are identical as long as the transformation satisfies the other constraints given in this section.

5.5 Design Space Exploration

More than one Galois NLFSR equivalent to a given Fibonacci NLFSR can in general be obtained. In general, the performances will vary between the different solutions because some will have longer feedback functions while others will have shorter ones. To explore the design space and find the best solution in terms of throughput, we used a modified version of the heuristic algorithm developed

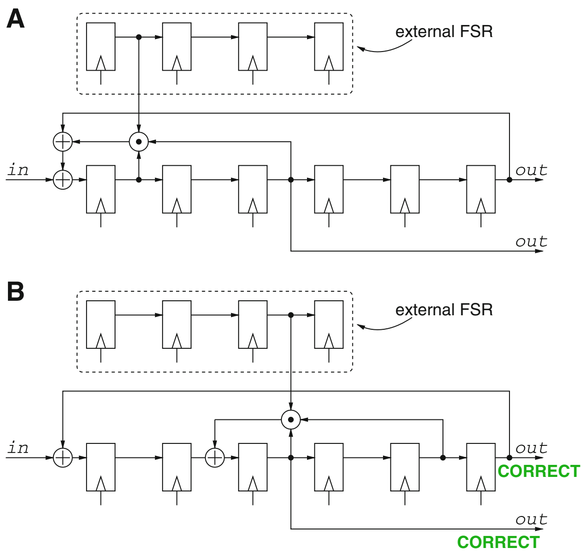


Fig. 7. FSR with external inputs: (A) Original Fibonacci FSR; (B) Transformed FSR

in [12]. The algorithm takes as input a Fibonacci FSR and a list of allowed update functions, i.e. update functions where the products can be placed. In output, the algorithm generates an efficient (high-throughput) Galois FSR. The algorithm was modified so that it supports shifting external signals along with internal ones.

The algorithm tries to place the product terms into the update functions in an efficient way, trying to minimize a *cost function*, which is an approximation of the critical path of the system.

6 U-Quark's Transformation

6.1 Structural Modification

The operation of U-Quark can be described as follows: the permutation block is initialized with the initial value and runs continuously. A counter counts continuously from 0 to 543 and then loops back to 0. When the counter value is 0 the L-LFSR state bits are resetted at the all-ones state. If the hash function is in the absorbing phase, the update functions $g_{67}, ..., g_{60}$ of the Y-NLFSR state bits $y_{67}, ..., y_{60}$ are substituted with the update functions $g^*_{67}, ..., g^*_{60}$, where the functions $g^*_{67}, ..., g^*_{60}$ are obtained by XORing the values of $g_{67}, ..., g_{60}$ with the next 8 message bits. The outputs are taken from the update functions $g_{67}, ..., g_{60}$ when the value of the counter is 0 and the message bits are finished (during the squeezing phase).

We will show that the functions on which U-Quark's Y-NLFSR product terms can be moved have grade between 67 and 59 and preliminarily we decide to use

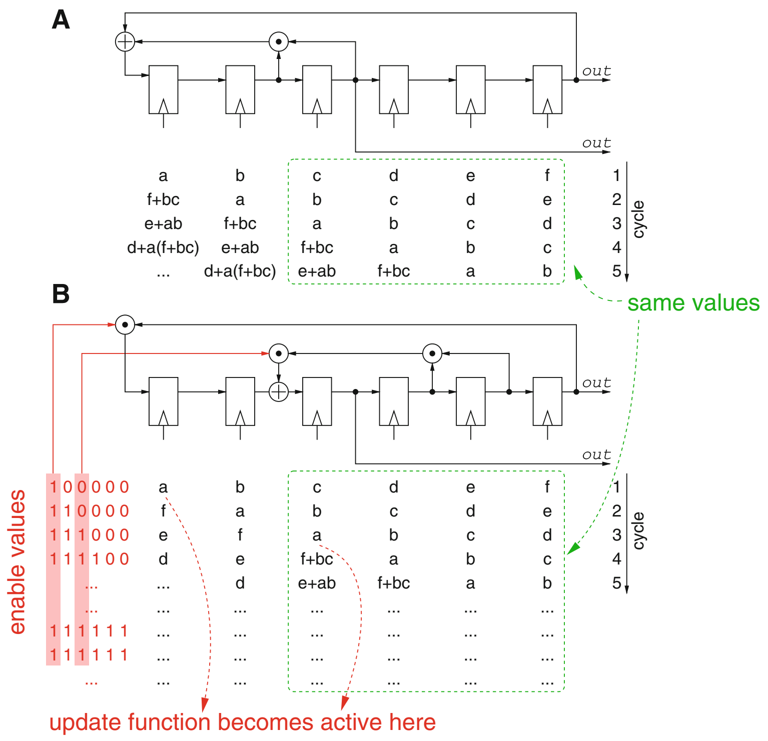


Fig. 8. (A) Original Fibonacci FSR; (B) Transformed parallel-loading FSR. In red are indicated the enable signals for the update functions. In this example, the only non-trivial update functions become active at cycles 1 and 3 respectively

all of the available feedback functions for product placements. This means that the lowest-grade non-trivial feedback function of the Y-NLFSR will be g_{59} and the original values of $g_{67}, ..., g_{60}$ are not available due to the considerations in Sect. 5 (only the state bits having grade lower than 59 match in the original and the transformed FSRs). However we observe that the values of $g_{67}, ..., g_{60}$ appear on the values $g_{58}, ..., g_{51}$ with a 9-cycle delay.

If the initialization value is programmable and can change from run to run, we transform U-Quark into the structure of Fig. 9: the hash function is initialized with the same value as for the original hash and then the system is clocked for 9 cycles turning on the feedback functions one-by-one, as discussed in Sect. 5. During the absorbing phase, when the counter value is 9, the update functions $g_{58}, ..., g_{51}$ of the Y-NLFSR state bits $y_{58}, ..., y_{51}$ are substituted with the update functions $g^*_{58}, ..., g^*_{51}$, where the functions $g^*_{58}, ..., g^*_{51}$ are obtained by XORing the values of $g_{58}, ..., g_{51}$ with the next 8 message bits. During the squeezing phase, the outputs are taken from the update functions $g_{58}, ..., g_{51}$ when the value of the counter is 9. The L-LFSR is reset to the all-ones state when the value of the counter is 0.

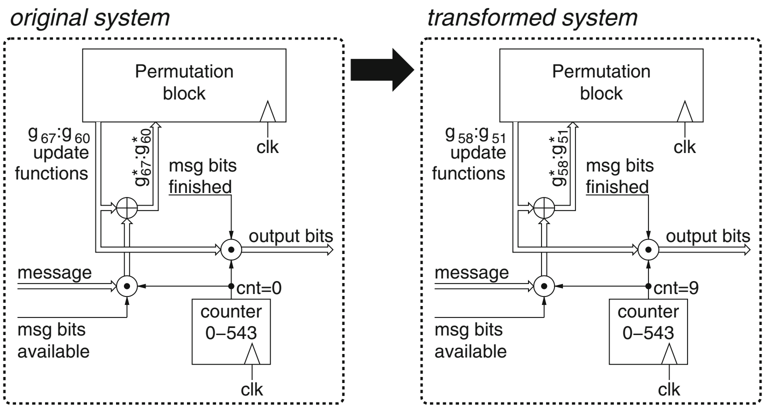


Fig. 9. (left) Original U-Quark; (right) Transformed U-Quark

If the initialization value is fixed and does not change, as is usually the case with low-weight hash functions, then we load the hash function with the initial value, load the L-LFSR with all ones, and run U-Quark offline (in a simulator) for 9 cycles, turning on the feedback functions one by one. The state of the permutation block after the 9th cycle is stored as the initial value into the hash function. The absorption of the new message bits and the squeezing of the digest bits happen both on $g_{58}, ..., g_{51}$ when the value of the counter is 0. This solution allows skipping the overhead of the block necessary to turn on the feedback functions one by one and the 9-cycles latency, since the cipher is clocked offline, in a simulator, for the first 9 cycles.

6.2 Updated X-NLFSR

For the X-NLFSR in U-Quark, the product term with the maximal difference in variable indexes is $x_9x_{28}x_{45}x_{59}$, i.e. $\tau_{min} = 50$. Product terms cannot be allocated to feedback functions x_i of grade $i < 56$ because bit x_{56} is used as an input in function $h(x, y, l)$. We decide to reserve the first three feedback functions for $h(x, y, l)$. Therefore, except for the product term x_0, which cannot be moved, we allocate all other product terms to feedback functions x_i of grade $i \leq 64$.

The following Galois NLFSR is obtained by the algorithm developed in [12] and described in Sect. 5.5 (for the values of h_{67}, h_{66} and h_{65} see Sect. 6.4):

$$\begin{aligned}
f_{67} &= x_0 \oplus y_0 \oplus h_{67} \\
f_{66} &= x_{67} \oplus h_{66} \\
f_{65} &= x_{66} \oplus h_{65} \\
f_{64} &= x_{65} \oplus x_{30}x_{34} \oplus x_{49} \oplus x_{34} \oplus x_{42} \oplus x_{47} \\
f_{63} &= x_{64} \oplus x_{51} \oplus x_{55}x_{51}x_{17}x_{11} \oplus x_{17} \\
f_{62} &= x_{63} \oplus x_{47}x_{40}x_{32}x_{27}x_{23}x_{16}
\end{aligned}$$

$$
\begin{aligned}
f_{61} &= x_{62} \oplus x_{53}x_{49}x_{46}x_{39}x_{31} \\
f_{60} &= x_{61} \oplus x_{30}x_{26}x_{45}x_{48} \oplus x_{38}x_{45}x_{48} \oplus x_2 \\
f_{59} &= x_{60} \oplus x_1x_{20}x_{37}x_{51} \oplus x_6 \oplus x_1x_7 \\
f_{58} &= x_{59} \\
f_{57} &= x_{58} \oplus x_{23}x_{18}x_{11} \oplus x_{49}x_{45} \\
f_{56} &= x_{57} \oplus x_{22}
\end{aligned}
$$

6.3 Updated Y-NLFSR

For the Y-NLFSR, the product term with the maximal difference in variable indexes is $y_7y_{30}y_{42}y_{58}$, i.e. $\tau_{min} = 51$. Product terms cannot be allocated to feedback functions y_i of grade $i < 59$ because bit y_{59} is used as a input in function $h(x, y, l)$. We decide to reserve the first three feedback functions for $h(x, y, l)$. Therefore, we allocate all other product terms to feedback functions x_i of grade $i \leq 64$.

The following Galois NLFSR is obtained by the algorithm developed in [12] and described in Sect. 5.5 (for the values of h_{67}, h_{66} and h_{65} see Sect. 6.4):

$$
\begin{aligned}
g_{67} &= y_0 \oplus h_{67} \\
g_{66} &= y_{67} \oplus h_{66} \\
g_{65} &= y_{66} \oplus h_{65} \\
g_{64} &= y_{65} \oplus y_{27} \oplus y_{32} \oplus y_{34} \oplus y_{39} \oplus y_{48} \oplus y_{51} \\
g_{63} &= y_{64} \oplus y_{47}y_{38}y_{33}y_{31}y_{26}y_{16} \oplus y_{50}y_{47}y_{33}y_{31} \\
g_{62} &= y_{63} \oplus y_2 \oplus y_{11} \oplus y_{53}y_{49} \oplus y_{32}y_{30} \oplus y_{10}y_2 \\
g_{61} &= y_{62} \oplus y_{48}y_{45}y_{36} \oplus y_{29}y_{24}y_{14} \oplus y_{52}y_{36}y_{24}y_1 \\
g_{60} &= y_{61} \oplus y_{51}y_{47}y_{13}y_8 \oplus y_{51}y_{47}y_{44}y_{35}y_{30} \oplus y_{28}y_{23}y_{13}y_8y_0 \\
g_{59} &= y_{60} \oplus y_{41} \oplus y_{50}y_{46}y_{43}y_{34}y_{29} \oplus y_{12}
\end{aligned}
$$

6.4 Updated *h* Function

We define the $h(x, y, l)$ function as h_{67} because it is fed to state bits x_{67} and y_{67} of the two NLFSRs. The $h_{x,y,l}$ function is split into the three functions h_{67}, h_{66} and h_{65}:

$$
\begin{aligned}
h_{67}(x, y, l) &= l_0 \oplus x_1 \oplus y_2 \oplus y_3x_{55} \oplus l_0x_{25}x_{46}y_{59} \oplus l_0x_{25} \\
h_{66}(x, y) &= y_2x_{45}x_{54} \oplus y_2x_{44}y_{58} \oplus y_2x_{21}x_{45} \oplus x_3 \oplus y_{58} \\
h_{65}(x, y) &= x_{44}x_{53} \oplus x_{53}y_{57} \oplus y_8 \oplus x_{23} \oplus x_{29} \oplus y_{41} \oplus x_{54}
\end{aligned}
$$

7 Implementation Results

Table 1 reports the maximal operating frequency of U-Quark after applying the transformation described in Sect. 6, as well as area and power figures. The improvements over the original hash function are also reported.

Table 1. Implementation results

	Frequency (GHz)	Area (μm^2)	Power (μW)
Original	1.86	4956	14.8
Optimized	2.50	5029	12.0
Improvement	34 %	0 %	19 %

Results were obtained by designing the hash functions at Register Transfer Level (RTL) in Verilog, and then synthesizing the code for best performances using Cadence RTL Compiler for the TSMC 90 nm ASIC technology. Power results were obtained by running gate-level simulation on the obtained netlist, using random test vectors and a 1 MHz operating frequency, and using the obtained toggle values to estimate the power consumption through the synthesis tool.

The transformed U-Quark has a 34 % higher throughput compared to the original Quark and consumes 19 % less power. We think that the power improvement is due to the shorter combinational paths in the transformed hash function, which allow the synthesis tool better optimization opportunities. The area of the original and the transformed hash functions are very close. The only drawback of the transformation is therefore the 9 cycles latency in the production of the hash, in case the initial value of the cipher is not fixed and can change from run to run.

8 Conclusion

In conclusion, we have shown that it is possible to considerably improve the hardware timing figures of the Quark hash function by applying a Fibonacci-to-Galois transformation of its feedback shift registers. We have extended the NLFSR Fibonacci-to-Galois transformation described in [7] so that it can support U-Quark's FSRs.

We could obtain a 34 % better throughput and a 19 % lower power consumption, without any area overhead. The transformation is very easy to apply because it only requires a modification of the feedback functions at RTL. If the initial state of the hash function is programmable and can change from run to run, a 9 cycles latency in the production of the hash is inserted. If the initial state is fixed, then no additional latency is inserted.

Acknowledgment. This work was supported in part the research grant No 621-2010-4388 from the Swedish Research Council and in part by the research grant No SM12-0005 from the Swedish Foundation for Strategic Research.

References

1. Aumasson, J.-P., Henzen, L., Meier, W., Naya-Plasencia, M.: QUARK: a lightweight hash. In: Mangard, S., Standaert, F.-X. (eds.) CHES 2010. LNCS, vol. 6225, pp. 1–15. Springer, Heidelberg (2010)

2. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: On the indifferentiability of the sponge construction. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 181–197. Springer, Heidelberg (2008)
3. Hell, M., Johansson, T., Maximov, A., Meier, W.: The grain family of stream ciphers. In: Robshaw, M., Billet, O. (eds.) New Stream Cipher Designs. LNCS, vol. 4986, pp. 179–190. Springer, Heidelberg (2008)
4. Hell, M., Johansson, T., Maximov, A., Meier, W.: A stream cipher proposal: Grain-128. In: 2006 IEEE International Symposium on Information Theory, pp. 1614–1618, July 2006
5. Agren, M., Hell, M., Johansson, T., Meier, W.: Grain-128a: a new version of Grain-128 with optional authentication. Int. J. Wire. Mob. Comput. **5**, 48–59 (2011)
6. De Cannière, C., Dunkelman, O., Knežević, M.: KATAN and KTANTAN — a family of small and efficient hardware-oriented block ciphers. In: Clavier, C., Gaj, K. (eds.) CHES 2009. LNCS, vol. 5747, pp. 272–288. Springer, Heidelberg (2009)
7. Dubrova, E.: A transformation from the Fibonacci to the Galois NLFSRs. IEEE Trans. Inf. Theory **55**(11), 5263–5271 (2009)
8. Mansouri, S.S., Dubrova, E.: An improved hardware implementation of the Grain-128a stream cipher. In: Kwon, T., Lee, M.-K., Kwon, D. (eds.) ICISC 2012. LNCS, vol. 7839, pp. 278–292. Springer, Heidelberg (2013)
9. Golomb, S.: Shift Register Sequences. Aegean Park Press, Laguna Hills (1982)
10. Mansouri, S., Dubrova, E.: An improved hardware implementation of the Grain stream cipher. In: 2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD), pp. 433–440, September 2010
11. Dubrova, E.: Finding matching initial states for equivalent NLFSRs in the Fibonacci to the Galois configurations. IEEE Trans. Inf. Theory **56**(6), 2961–2967 (2010)
12. Chabloz, J.-M., Mansouri, S.S., Dubrova, E.: An algorithm for constructing a fastest Galois NLFSR generating a given sequence. In: Carlet, C., Pott, A. (eds.) SETA 2010. LNCS, vol. 6338, pp. 41–54. Springer, Heidelberg (2010)

Analyzing Side-Channel Leakage of RFID-Suitable Lightweight ECC Hardware

Erich Wenger(✉), Thomas Korak(✉), and Mario Kirschbaum

Institute for Applied Information Processing and Communications,
Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria
{Erich.Wenger, Thomas.Korak, Mario.Kirschbaum}@iaik.tugraz.at

Abstract. Using RFID tags for security critical applications requires the integration of cryptographic primitives, e.g., Elliptic Curve Cryptography (ECC). It is specially important to consider that RFID tags are easily accessible to perform practical side-channel attacks due to their fields of applications. In this paper, we investigate a practical attack scenario on a randomized ECC hardware implementation suitable for RFID tags. This implementation uses a Montgomery Ladder, Randomized Projective Coordinates (RPC), and a digit-serial hardware multiplier. By using different analysis techniques, we are able to recover the secret scalar while using only a single power trace. One attack correlates two consecutive Montgomery ladder rounds, while another attack directly recovers intermediate operands processed within the digit-serial multiplier. All attacks are verified using a simulated ASIC model and an FPGA implementation.

Keywords: Implementation attack · Correlation power analysis · Simple power analysis · Digit-serial multiplier · Elliptic curve cryptography

1 Introduction

When it comes to RFID security research, many research groups all around the world investigate the viability of new protocols, new optimized algorithms, and new hardware implementation for RFID tags. Those designs have to cope with the restrictive area, power, and runtime challenges that are mandatory for practically usable RFID tags. Additionally, also power-analysis attacks have to be considered. Those attacks can be used to recover keys, even though the actual protocol or algorithm is mathematically secure.

The most promising public-key protocols are based on Elliptic Curve Cryptography (ECC) as ECC offers comparably small memory and practically useable runtime properties. RSA and ElGamal based public key schemes simply need too much memory or only provide inferior runtimes. Some of the most notable ECC implementations are [5,12,22,23,33,39]. Unfortunately, there have been too few practical evaluations of those state-of-the-art hardware implementations. Especially the design of Lee et al. [23] raised our interest. They use a digit-serial multiplier with a López and Dahab-like [24] Montgomery Ladder. Further we

M. Hutter and J.-M. Schmidt (Eds.): RFIDsec 2013, LNCS 8262, pp. 128–144, 2013.
DOI: 10.1007/978-3-642-41332-2_9,

assume that the design-under-attack performs in constant key-independent runtime, utilizes Randomized Projective Coordinates [9] (RPC), and use the private scalar only once (as it is done for ECDSA signatures and Diffie-Hellman key exchanges). Therefore we have very strong assumptions regarding the actual implementation under investigation and many of the related power analysis techniques [7,10,17,20,21,26] simply cannot be mounted.

Our contribution. In this paper, we successfully perform simple and correlation-based power analysis attacks on a protected ECC hardware implementation using only a single power trace. The investigated hardware design utilizes a López and Dahab Montgomery ladder, RPC, ephemeral secret keys, a digit-serial binary field multiplier, and performs in constant runtime. We show the practicability of our attacks using simulated and FPGA-measured power traces. The correlation based attack shows how the hamming distance of consecutive key bits can be used to recover the secret scalar. Additionally, we thoroughly analyze the power consumption of bit-serial and digit-serial binary field multiplier. We are able to recover one of the processed operands and discuss how to utilize this information to perform more advanced attacks.

The paper is structured as follows: Section 2 discusses related work. Sections 3–5 elaborate the design under attack, some attack-related prerequisites, and the basic setup for conducting the attacks, respectively. In Sect. 6 we assure that there is no key-dependent leakage. Section 7 discusses the correlation of consecutive rounds and Sect. 8 discusses the recovery of intermediate values. Section 9 concludes the paper.

2 Related Work

There exist many papers that describe hardware implementations of ECC. Most of them make use of digit-serial multipliers over $GF(2^m)$, see for example [1,4,5, 11,14,23,30]. The reason for that choice lies in several facts. First, binary-field multipliers significantly improve the performance of ECC since they avoid carry-propagation as opposed to prime-field based implementations. Second, since they are based on binary polynomials, they can be efficiently implemented in hardware which makes them especially attractive for embedded systems and low-area designs. Therefore, they are commonly used and applied in real-world applications such as contactless smart cards, RFIDs, or Java Cards [1,6,29].

In view of side-channel attacks, there exist many papers that discuss attacks and countermeasures on ECC, for example presented in [8,9,13,18,27,31,32,36, 37]. Most of the work exploits the weakness of different scalar-multiplication algorithms such as double-and-add which allows to perform SPA attacks in order to distinguish between a single *double* or *add* operation.

Most notable is the work of Walter [36], who attacked RSA using only a single trace. In a sliding-window RSA multiplication, he correlates the preprocessing step with the per-bit multiplication. He notes that such an attack can even work using a single power trace. The attack on the RSA modular exponentiation by Wittemann et al. [38] can be performed even in the presence of the message blinding and the multiply always countermeasures. They take advantage of the fact that multiplications followed by squarings share operands in a key-dependent manner. In order to extract the bits of the secret exponent one power measurement recorded during the exponentiation is sufficient. In 2010, Clavier et al. [8] introduced the terms vertical and horizontal power analysis. While a vertical power analysis attacks the same time sample on many curves, a horizontal power analysis correlates parts of a single power trace. They performed a horizontal correlation analysis on RSA and similar to Amiel et al. [3] distinguished multiplication from squaring operations. Also in 2010, Homma et al. [18] generated collisions between squaring operations by recording only two power traces.

While those attacks work on the group structure of RSA and ECC, there exist only a few papers that demonstrate the susceptibility of underlying finite-field arithmetics. In 2006, Akishita et al. [2] demonstrated an attack on ECC by measuring the difference of modular multiplication and squaring. Since they performed attacks targeting ECC-field operations instead of ECC-group operations, their attack is applicable to countermeasures such as unified-addition formulae or SPA-resistant algorithms like the Montgomery-powering ladder. Recently, Pan et al. [34] presented a correlation power-analysis (CPA) attack on a digit-serial multiplier. They targeted the output register of the multiplier and successfully extracted intermediate values from an FPGA implementation. However, since they applied a CPA attack using 1,000 traces, their attack cannot be applied on ECC implementations that use random scalars.

In the following, we present a power-analysis attack that extracts the secret scalar from an ECC implementation consisting of a Montgomery ladder and RPC using only one single power trace. The attack can therefore be even applied to reveal ephemeral keys such as used in the Elliptic Curve Digital Signature Algorithm (ECDSA) or in Elliptic Curve Diffie Hellman (ECDH) protocols.

3 Design Under Attack

In the following, the ECC implementation used for the conducted experiments is presented. The objective of the implementation is to provide resistance against side-channel attacks as well as being flexible in size and runtime. The SCA resistance is achieved by using a constant-runtime Montgomery ladder with randomized projective coordinates and the flexibility in size and runtime is achieved by using a digit-serial multiplier.

3.1 Chosen Top-Level Algorithms

Under the consideration and investigation of related work on timing, power-analysis, and fault attacks applicable on elliptic curve cryptography, we decided to use a left-to-right Montgomery ladder for EC point multiplications $Q \leftarrow k \times P$. The key-independent structure of the Montgomery ladder provides a good foundation against many side-channel attacks. In order to be independent of the most significant bits of the scalar k, the order of the elliptic curve is added to k. Additionally by randomizing the projective coordinates of the base-point, most attacks become infeasible. So a combination of a constant-runtime Montgomery ladder with randomized projective coordinates provides strong resistance against most side-channel attacks.[1]

3.2 Hardware Design

Similar to Lee et al. [23], our hardware is specially optimized for binary extension fields using the NIST [28] standardized elliptic curve B-163. The hardware design comes with a two-port 163-bit memory, a 163-bit adder, and an MSB-first digit-serial multiplier. In order to save chip area, no dedicated hardware squaring unit is used. For point multiplication we use the formulas by López and Dahab [24] (see Appendix A).

3.3 Digit-Serial Multiplier

Most critical for the size of the hardware design, the runtime of the design and the following attacks is the multiplier used in the design. In the following we give a brief introduction to digit-serial multipliers, which are used in our design.

The term "digit serial" indicates that only a limited number of digits d of operand OpB are processed in each clock cycle. In the special case of $d = 1$, the multiplication approach is also known as bit-serial multiplication approach. A useful property of the digit-serial multiplication approach is that it can be sped up by increasing the digit size d. Thus, the designer can easily change the design to meet the desired area and runtime constraints.

Digit-serial multipliers are used to calculate the product $C \leftarrow OpA \times OpB$, where N bits are required to represent OpA, OpB and C. In the remainder of this paper, we use the notation OpB_i to index a single digit with index i, with $0 \leq i < \lceil N/d \rceil$ and $\lceil N/d \rceil - 1$ indexing the most significant digit. Figure 1 shows two approaches for bit-serial multiplications ($d = 1$) using a fixed reduction polynomial (cf. [15]). Either the most significant bit (MSB) is processed first (shown on the left) or the least significant bit (LSB) is processed first (shown on the right). Since for the LSB-first multiplier, two registers (a_i and c_i) are modified in each cycle, we concentrate on the MSB-first multiplier. The single active (working) register C (respectively c_i) is shifted by d digits to the left and is implicitly reduced using a fixed reduction polynomial. Additionally, OpA (resp. a_i) is multiplied with OpB_i using a simple AND gate. The product of the multiplication is then added to the (by d bits) shifted and reduced work register. After $\lceil N/d \rceil$ cycles, the product ($C \leftarrow OpA \times OpB$) of the finite-field multiplication is stored in register C.

This multiplication approach can be applied to binary-extension fields as well as prime fields. For prime fields, the XOR-gates have to be replaced with

[1] Note that we are aware of fault attacks, but those type of attacks are not subject of this paper.

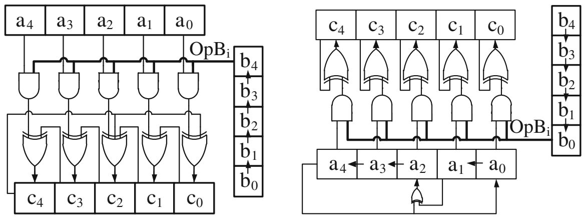


Fig. 1. Finite-field multiplier with fixed reduction polynomial $f(z) = z^5 + z^2 + 1$.

full adders. Thus digit-serial multipliers can be used for RSA as well as Elliptic Curve Cryptography.

4 Attack Prerequisites

In order to perform the attacks presented in the following sections, the design has to fulfill some prerequisites which are discussed here.

4.1 Constant Runtime Scalar Multiplication

After recording one power trace while the device under attack performs the scalar multiplication a post-processing step is necessary. The trace T is split in $|k|$ traces after removing the initialization (t_{init}) and final y-recovery phases. Here we take advantage of the fact that all round transformations have an equal runtime. For simplification purposes we assume that $|k| = N$. Sub-traces $R_i = T(t_{init}+t_{round}(N-1-i) \dots t_{init}+t_{round}(N-i))$ represent a López-Dahab double-and-add round operation of the secret scalar k_i, with $i = [0, N-1]$. According to our notation k_{N-1} represents the MSB and k_0 the LSB of k. One method to identify the length of one round (t_{round}) is to perform a cross correlation on the trace T. The result of the cross correlation shows significant, equidistant peaks. The distance corresponds to t_{round}. In the following $R_i(o, o+w)$ denotes a part of the trace of round i with an offset o from the beginning and a length of w samples.

Figure 2 shows a comparison of the simulated and measured power consumption with $d = 1$. For the figures the traces $R_0 \dots R_{N-1}$ are plotted overlayed. Apparently both traces show identical characteristics. The left half of the traces (cycles 995-1150) shows a multiplication of pseudo-random OpA with pseudo-random OpB. In the right half (cycles 1150-1320) OpB has been kept unchanged during the N round transformations. In fact a multiplication with the constant $c = b^{2^{m-1}} \bmod f(z)$ which is used within the López-Dahab formula [24] is performed in this interval. This unchanged operand leads to a similar power consumption in this interval for all round transformations.

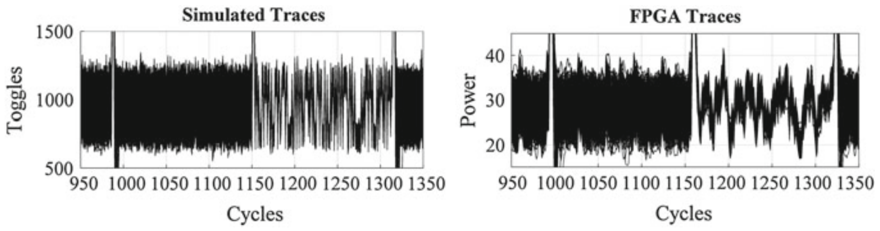


Fig. 2. Comparison of simulated traces (on the left) and traces recorded on the FPGA (on the right).

4.2 Leakage of the Digit-Serial Multiplier

With the field multiplication being the most time-consuming part of an EC point multiplication and the digit-serial multiplier being the most active part of the circuit, the side-channel vulnerability is highly dependent on the power-consumption of a digit-serial multiplier.

The main source of leakage within the digit-serial multiplier lies within the multiplication of the full word OpA with a single digit OpB_i. In a first model (which was wrong) we theorized that there will be a higher power consumption when $OpB_i = 1$ (with $d = 1$) and a lower power consumption with $OpB_i = 0$. Such a power model may be true for a logic style such as Transistor-Transistor-Logic (TTL), but for a CMOS process our power model had to be refined. In CMOS, the power consumption does not depend on the current state of a signal, but on the transition from the previous state to the next state. So the power consumption is related to the Hamming distance of two consecutive values of OpB_i. Let $w(\cdot)$ denote the Hamming weight and $hd(\cdot, \cdot)$ denote the Hamming distance. Experiments showed that the higher the Hamming distance $hd(OpB_i, OpB_{i+1})$, the higher is the power consumption of the circuit. Let $hd(OpB)$ be a vector with all $hd(OpB_i)$ and $hd(OpB_i)$ be a short form for $hd(OpB_i, OpB_{i+1})$.

Before we discuss the impact of the digit-serial multiplier on the side-channel leakage in Sect. 8 in detail, we must elaborate our basic side-channel analysis approach and related assumptions.

5 Setup for Conducting the Attacks

Two approaches to record the required power trace for the performed attacks are used. On the one hand the power traces are generated using a simulation of the implementation. On the other hand the design was implemented on an FPGA and the power consumption was measured using an oscilloscope. The achieved results were compared and prove the correctness of the attack assumptions.

5.1 Simulator Toolchain

A simulator toolchain was used in order to enable an attack on the ECC implementation in a noise-free environment. This toolchain consists of four elements:

Cadence RTL Compiler v08.10 was used to synthesize the VHDL code to UMC L-130 logic gates; Cadence First Encounter v08.10 was used to place and route the design; NCSim v08.20 was used to simulate the routed design and generate a value-change-dump (VCD) file; and a VCD analyzer was used to generate simulated power traces from the VCD file by applying a toggle-counting technique. By accumulating all toggles within a clock cycle, the resulting trace contained one data value per clock cycle. As shown in Kirschbaum et al. [19], simulated power traces derived from toggle counts are well comparable to SPICE power simulations as well as to real power measurements of an integrated circuit.

5.2 FPGA Measurement Setup

We furthermore synthesized the design (with $d = 1$) on an FPGA. This step enables us to record the power consumption during the targeted point multiplication performed on a real-world device. As FPGA a Virtex-II Pro xc2vp7 was used which is part of the SASEBO [35] side-channel evaluation board. As the development environment, Xilinx 10.1.03 was used. For recording the power traces we have used a LeCroy WP725Zi oscilloscope with a sampling rate of 2.5 GS/s and operate the device under test at a clock rate of 25 MHz. Our first measurements showed that for the attacks an accurate clock frequency is beneficial in order to avoid the introduction of noise due to clock jitter. Using an off-the-shelf quartz increases the effort for the attack as additional postprocessing steps on the recorded trace are required. In particular, a fixed number of samples per clock period is required for the attack, that means the ratio between the sampling rate and the clock rate must be integer. If this is not the case an upsampling of the recorded trace is required in order to achieve this ratio. To circumvent this we used an Agilent 33250A signal generator as external clock source. Further the complete point multiplication must be finished before the used oscilloscope runs out of "input buffer". Our LeCroy WP725Zi oscilloscope is capable of storing 64 million samples per trace.

In order to decrease the calculation and memory effort we performed a downsampling step on the recorded trace. In this preprocessing step, all the sample points within one clock period of the device were summed up. Once again, we want to emphasize that the following attacks work using only *a single power trace* of the scalar multiplication.

6 Assuring Side-Channel Resistance

In order to make sure that our implementation has no key dependent leakage, we performed a basic difference-of-means analysis with a known scalar. By redesigning the hardware, all sources of leakage were eliminated. The difference-of-means trace in Fig. 3 shows the leakage of a preliminary version of our hardware design under investigation.

Theory. To launch a difference-of-means attack (cf. [25]), all key-dependent round traces are categorized according to k_i. Next, the average $avg(R_i|_{k_i,\forall i})$

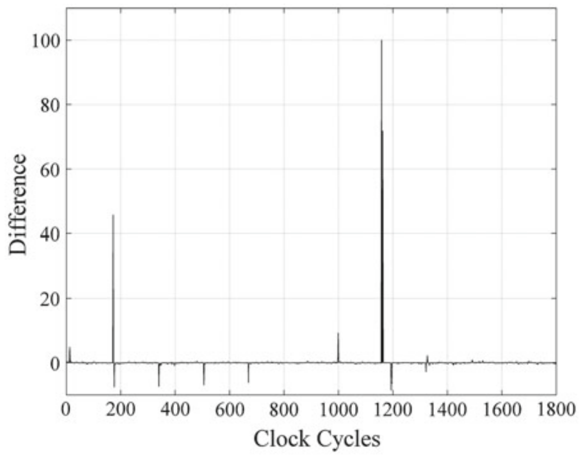


Fig. 3. Difference-of-means of a simulated power trace.

and the difference of the means in the respectable categories $avg(R_i|_{k_i=0,\forall i}) - avg(R_i|_{k_i=1,\forall i})$ are calculated. By investigation of the resulting plot it is possible to find key-dependent operations in the trace.

Practical Results. A difference-of-means calculation was performed using the subtraces R_i extracted from a single measured power trace from an early FPGA implementation and is depicted in Fig. 3. The operations at clock cycles with a high difference show some key-dependent behavior. The higher the difference is, the easier it is for an attacker to find the key-dependent parts. This results allow a designer to identify key-dependent operations and improve the design. In our special case we found that at clock cycle 1160, the access to the data memory was dependent on the key bit k_i. For all subsequent experiments this error was obviously fixed.

We redesigned our hardware implementation until the most significant peaks of the difference-of-means analysis vanished. Therefore we covered the basics and assured the significance of the following power analysis attacks.

7 Correlation of Consecutive Rounds

The assumption is that in consecutive rounds R_i and R_{i-1} similarities dependent on the processed bits k_i and k_{i-1} can be found. This holds for our implementation using the formulas of López and Dahab (cf. Appendix A) for point multiplication but is also applicable to other algorithms.

Theory. In this scenario we focus on finding similarities in the power traces R_i and R_{i-1} of two consecutive rounds. Once again it has to be mentioned that these power traces are subtraces of equal length of one power trace recorded during the point multiplication. The idea behind this approach is that the power profile of a digit-serial multiplication is mostly dependent on OpB. If the same operand OpB has been used in two consecutive rounds similarities can be found. Witteman et al. [38] have used a similar assumption in their attack on the RSA modular exponentiation. They take advantage of the fact that the square-and-multiply-always algorithm reuses operands in a key-dependent manner. As similarity measure we have tried the correlation coefficient as well as the Euclidean distance. Both approaches lead to comparable results with runtime advantages for the Euclidean distance. Equation 1 shows how to calculate the Euclidean distance of two consecutive rounds with offset o_i, o_{i-1} respectively and a window size w. In Eq. 2, the formula to generate the distance matrix for the rounds i and $i-1$ with the given limits for j and l can be found.

$$d(R_i, R_{i-1})|_{o_i,o_{i-1}} = ||R_i(o_i, o_i + w), R_{i-1}(o_{i-1}, o_{i-1} + w)|| \tag{1}$$

$$Dist_l^j(R_i, R_{i-1}) = d(R_i, R_{i-1})|_{j,l} \tag{2}$$
$$|k| \geq i \geq 1; 0 \leq j \leq t_{round} - w; 0 \leq l \leq t_{round} - w;$$

Practical Results. Figure 4 shows the results of a windowed correlation of two consecutive rounds with a window size $w = 163$ using the traces recorded from the FPGA. The length of the subtraces R_i is 1796 samples. Small Euclidean distances are indicators for similar intermediate values. Those traces with small Euclidean distances have been highlighted. Two cases are distinguishable in Fig. 4. In the first case (e.g., between rounds 160 and 159) three locations with a small Euclidean distance can be identified. In the second case (e.g., between rounds 161 and 160) only one location with a small Euclidean distance can be identified. By investigating the formulas of López and Dahab [24], we identified the peak around offset 1150 as $OpB = c = b^{2^{m-1}} \mod f(z)$. This single peak appears in all correlation figures at the same position. The other two peaks appear, when the Hamming distance $hd(k_i, k_{i-1}) = 1$. In other words: the peaks appear,

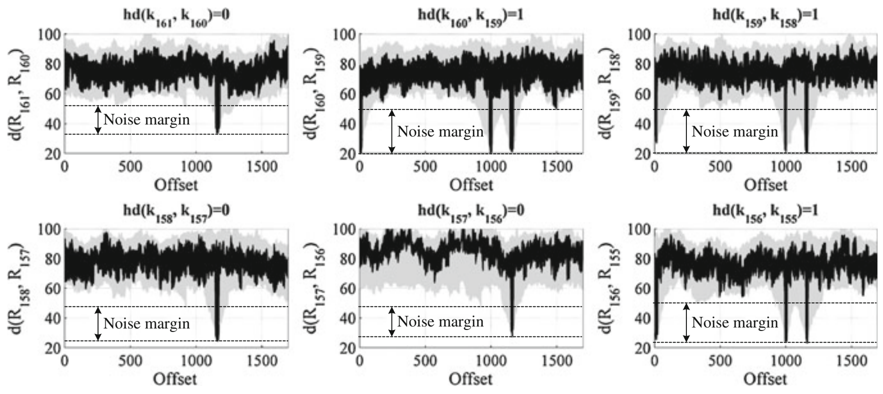


Fig. 4. Euclidean distance traces.

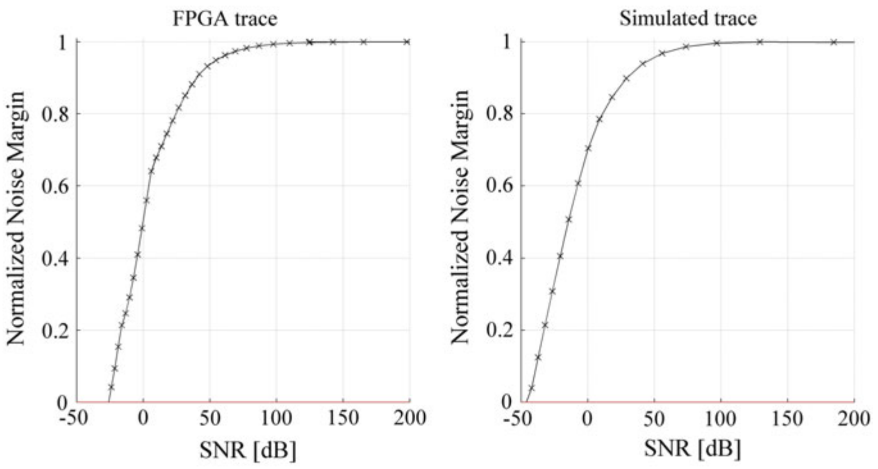


Fig. 5. Influence of noise on the noise margin.

when the key bit k_i is different from k_{i-1}. In such a case one operand OpB from round i is used again (unchanged) in round $i-1$. For a better understanding of the underlying problem, we attached the used formulas in Appendix A.

We also performed the same attack on the simulated traces and achieved comparable results. As the simulated traces do not contain measurement noise, the minimum Euclidean distances are even smaller. The presented attack worked for $d = 1$, $d = 2$, and $d = 4$ (used window sizes: $w = 163$, $w = 82$, and $w = 41$ respectively).

In order to visualize the influence of noise (e.g. introduced by the measurement environment or active noise countermeasures), simulated gaussian noise with different power levels has been added to the simulated as well as the measured trace. Figure 5 shows the normalized noise margin as a function of the signal to noise ratio SNR. The SNR has been calculated according to Eq. 3. The evolution of the noise margin for simulation and FPGA experiment is similar, only the value of the SNR where the noise margin comes below zero is different. For the FPGA experiment the noise margin comes below zero at −24 dB and for the simulation at −43 dB. The difference can be explained by the fact that the trace recorded with the FPGA already contains some noise. The trace extracted using the simulation on the other hand can be seen as noise free.

$$\mathrm{SNR} = 20 \cdot \log \frac{U_{\mathrm{eff,Signal}}}{U_{\mathrm{eff,Noise}}} \tag{3}$$

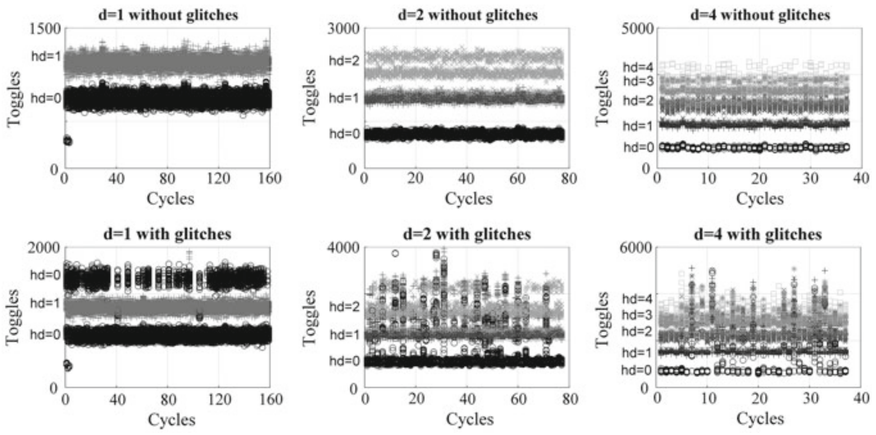


Fig. 6. Simulated power profiles for different d in the presence of glitches.

8 Revealing Intermediate Operands

In this section, we want to uniquely identify the intermediate finite-field polynomials used as operand OpB and discuss some more advanced attack scenarios. For a better understanding of the associated challenges, let us first investigate practical simulations.

Assumptions and Transferability. For this scenario we need to assume that a digit-serial multiplier is used. It is applicable to all designs based on digit-serial multipliers.

Practical Results. Figure 6 shows simulated toggle counts for $|k|$ overlayed N-bit multiplications. Using reference simulations, we marked the different Hamming distances $hd(OpB_i)$ with respective symbols. The following three characteristics are eye-catching: The different Hamming distances are distinguishable. The larger the digit size d is, the less distinguishable are the different levels of power consumption. And the more glitches occur on OpB_i, the less separable are those power levels. In our hardware design, we used an N-bit wide 5:1 multiplexer and a N:d multiplexer to select OpB_i. The resulting glitches are clearly visible in the lower row of Fig. 6. The upper row is the result of a d bit register being added after the multiplexers to suppress those glitches.[2] Also if the multiplexers are replaced by shift registers, there are virtually no glitches.

Interestingly, in the case of $d = 1$, glitches only occurred in a $1 \rightarrow 1$ transition. Obviously, the AOI-gates used in our design suppress glitches in $0 \rightarrow 0$ transitions.

Identifying the Intermediates. At this point, the attacker wants to identify the intermediate values processed during the field multiplications by using the

[2] For high-performance ECC implementations those registers are necessary in order to achieve the desired timings.

Table 1. Average number of solutions for OpB assuming $hd(OpB)$ is given.

Parameter	N =163	N =256
$d = 1$	$2^1 = 2^1$	$2^1 = 2^1$
$d = 2$	$2^2 2^{0.5\times 81} = 2^{42.5}$	$2^2 2^{0.5\times 127} = 2^{65.5}$
$d = 3$	$2^3 3^{0.75\times 54} = 2^{67.2}$	$2^3 3^{0.75\times 85} = 2^{104}$
$d = 4$	$2^4 4^{0.5\times 40} 6^{0.375\times 40} = 2^{82.8}$	$2^4 4^{0.5\times 63} 6^{0.375\times 63} = 2^{128.1}$

given power traces. The identification of the operand(s) OpB_i has to be done in two phases:

At first, the leaking Hamming distances in the power trace have to be classified. By overlaying all $|k|$ rounds, as it has been done in Fig. 6, it is possible to distinguish clusters of different Hamming distances. This classification can for instance be performed using a k-means algorithm [16]. We performed this classification on our simulated power traces and were able to detect the correct Hamming distances with the following error rates: 0.33 % for $d = 1$ without and with glitches, 4.13 % for $d = 2$ without glitches, 9.31 % for $d = 2$ with glitches, 4.96 % for $d = 4$ without glitches and 9.04 % for $d = 4$ with glitches.

During the second phase, we used the Hamming distances $hd(OpB)$ to iterate through all possible solutions. Several possible solutions for OpB result in the same $hd(OpB)$. Equation 4 gives an average number of possible solutions $N_{solutions}$ in dependence of N and d, when $hd(OpB)$ is given. $\#(hd = h)$ is the number of possible solutions for a fixed Hamming distance h and $p(hd = h)$ is the probability of the Hamming distance h. Since it is necessary to guess OpB_{MSB}, 2^d is a multiplicand factor within the equation. Table 1 gives exemplary numbers for $N = 163$, $N = 256$, and $d = 1...4$.

$$N_{solutions} = 2^d \cdot \left(\prod_{h=0}^{d} \#(hd = h)^{p(hd=h)} \right)^{\lceil \frac{N}{d} \rceil - 1} \tag{4}$$

The following short example should clarify the problematic: When $d = 2$, $hd(OpB_i, OpB_{i+1})$ is either 0, 1, or 2. Whereas $hd = 0$ and $hd = 2$ uniquely map OpB_i to a single OpB_{i+1}, $hd = 1$ does not. Let us assume $OpB_i = 00_b$. Then there are two solutions $hd(00_b, 01_b) = hd(00_b, 10_b) = 1$. With our power model,[3] those two solutions cannot be distinguished in the power trace. In average there are $2^{42.5}$ possible solutions (cf. Table 1) for $d = 2$ and $N = 163$. Using brute-force, breaking $2^{42.5}$ is practicable.

Other exemplary cases such as $d > 2$ or $N = 256$ are theoretically possible but impractical. However Table 1 depicts an *average* case. In practice the number of possibilities $N_{solutions}$ for a certain OpB depends on $hw(OpB)$. Hence, it is clever to only attack OpB with a small search space. An approximation of the necessary runtime can be performed in constant time. Furthermore we are confident that by investigating the different arrival times of single bits in OpB_i and by using more detailed timing information, our leakage model can be refined.

[3] Note that this might be possible using more advanced power models.

For many practical implementations without point randomization, finding those intermediate values would be sufficient. However, in the context of this paper (with point randomization) finding the intermediate values is only a preparational step for the following attack scenarios.

8.1 Correlate with an Arithmetic Combination of Intermediates

In Sect. 7, a direct correlation of the power trace on two consecutive rounds was performed. By changing the order of the operands of the multiplication this attack can be prevented. In many cases there is still a link between consecutive rounds when the result of an arithmetical combination of several operands $OpB^1, OpB^2, \ldots$ in round i is used as operand in round $i+1$. If the arithmetical combination $f(\cdot)$ as well as a set of operands $OpB^1, OpB^2, \ldots$ for round i is known, a set of used operands $F = f(OpB^1, OpB^2, \ldots)$ for round $i+1$ can be calculated. Feeding $f(\cdot)$ with every possible combination of $OpB^1, OpB^2, \ldots$ and performing the correlation $d(hd(F), R_{i-1})$ the size of the possible key candidates can be decreased. The complexity of this attack highly depends on the number of operands N_{OpB} used as arguments for $f(\cdot)$ as well as on d. N_{OpB} as well as d have an influence on the number of possible results for F. If e.g. $f(\cdot)$ is a squaring function ($N_{OpB} = 1$) and $d \leq 2$ the attack can be performed within acceptable bounds. Furthermore it has to be considered that if some bits in F are wrong there might still be a significant peak in the correlation plot, so there is no need to find the exact value of F. This fact also decreases the attack complexity.

8.2 Attack Several Intermediates Simultaneously

An enhancement of the previous scenario is to attack $(OpB^1, OpB^2, \ldots)$ as well as $F = f(OpB^1, OpB^2, \ldots)$ simultaneously. Let us assume the Hamming distances of F and $OpB^1, OpB^2, \ldots$ are known, so only a limited number of combinations for F and $OpB^1, OpB^2, \ldots$ fulfill the equation $F = f(OpB^1, OpB^2, \ldots)$. Although we did not test it, we are confident that by attacking different intermediates simultaneously the search space $N_{solutions}$ can be reduced by a certain degree.

8.3 Find the x-Coordinate

If an attacker can reveal the exact values for X_i (projective X coordinate in round i) and Z_i, she can calculate the x-coordinate of the currently processed point. Even if X_i and Z_i have been randomized and multiplied with a random λ this attack works. In the case of $X_r = X_i \cdot \lambda$ and $Z_r = Z_i \cdot \lambda$,

$$x_i = X_r \cdot Z_r^{-1} = (\lambda X_i) \cdot (\lambda Z_i)^{-1} = X \cdot Z^{-1} \quad (5)$$

can be recovered. Assuming a scalar multiplication $Q = k \times P$ is performed, small multiples of P can be precalculated and compared with x_i which is revealed in each round. Thus step by step all bits of k can be revealed. Even if there is an error in round i and x_i cannot be matched, x_{i+1} can be used to identify more key bits at once.

8.4 Undo the Projective Coordinate Randomization

In order to perform a projective coordinate randomization, each coordinate is multiplied with a random number λ. For that operation usually the same finite-field multiplier is used as the finite-field multiplier used during a point multiplication. So in the case of $X_r = X \cdot \lambda$ and λ is used as OpB, the randomization factor can be found. By knowing λ many attack scenarios on a device doing a point multiplication become feasible.

9 Conclusion

Nowadays the community knows that no implementation is believed to be secure as long as it has not been attacked and investigated in sufficient detail. In this paper, we attacked a protected ECC implementation by using only a single power trace. So even the popularly used ECDSA and the Diffie-Hellman key exchange algorithms are vulnerable. The corollary one should take away from this paper is that a designer must not only consider a simple difference-of-means attack but also be aware of more advanced and unexpected attack scenarios such as the here shown custom correlation attack or the attack on all processed intermediate values. But how should any designer be aware of future, currently unknown attacks?

Acknowledgments. The research described in this paper has been supported, in parts, by the European Commission through the ICT Program under contract ICT-SEC-2009-5-258754 TAMPRES, and by the Austrian Research Promotion Agency (FFG) and the Styrian Business Promotion Agency (SFG) under grant number 836628 (SeCoS).

Appendix A: Used Double-And-Add Formula

In this paper we used a slight modification of the Montgomery Ladder by López and Dahab. Algorithm 1 shows three iterations of the used double-and-add algorithm for three consecutive key bits. The modification from the original formula can be found in line 6. Here we swapped the order of x and Z_1, which is allowed according to the law of commutativity. During the first two iterations (left and middle column), an identical key bit is handled. So there is only a correlation when the constant c is used. During the second and third iteration, in which the key bit differs, Z_1 is used multiple times. Consequently in Fig. 4 three easily distinguishable peaks occur. A correlation of c and Z_1 can be observed.

Algorithm 1 López and Dahab round operations with key bits (0-0-1).

```
Ensure: $P_1' \leftarrow P_1 + P_2$.
Ensure: $P_2' \leftarrow 2 \cdot P_2$.
    Point Addition
 1: $X_1 \leftarrow X_1 \cdot Z_2$
 2: $Z_1 \leftarrow Z_1 \cdot X_2$
 3: $T_1 \leftarrow X_1 \cdot Z_1$
 4: $Z_1 \leftarrow Z_1 + X_1$
 5: $Z_1 \leftarrow Z_1 \cdot Z_1$
 6: $X_1 \leftarrow x \cdot Z_1$
 7: $X_1 \leftarrow X_1 + T_1$
    Point Doubling
 8: $X_2 \leftarrow X_2 \cdot X_2$
 9: $Z_2 \leftarrow Z_2 \cdot Z_2$
10: $T_1 \leftarrow Z_2 \cdot \boxed{\mathbf{c}}$
11: $Z_2 \leftarrow Z_2 \cdot X_2$
12: $T_1 \leftarrow T_1 \cdot T_1$
13: $X_2 \leftarrow X_2 \cdot X_2$
14: $X_2 \leftarrow X_2 + T_1$
```

```
Ensure: $P_1' \leftarrow P_1 + P_2$.
Ensure: $P_2' \leftarrow 2 \cdot P_2$.
    Point Addition
 1: $X_1 \leftarrow X_1 \cdot Z_2$
 2: $Z_1 \leftarrow Z_1 \cdot X_2$
 3: $T_1 \leftarrow X_1 \cdot Z_1$
 4: $Z_1 \leftarrow Z_1 + X_1$
 5: $Z_1 \leftarrow Z_1 \cdot Z_1$
 6: $X_1 \leftarrow x \cdot \boxed{\mathbf{Z_1}}$
 7: $X_1 \leftarrow X_1 + T_1$
    Point Doubling
 8: $X_2 \leftarrow X_2 \cdot X_2$
 9: $Z_2 \leftarrow Z_2 \cdot Z_2$
10: $T_1 \leftarrow Z_2 \cdot \boxed{\mathbf{c}}$
11: $Z_2 \leftarrow Z_2 \cdot X_2$
12: $T_1 \leftarrow T_1 \cdot T_1$
13: $X_2 \leftarrow X_2 \cdot X_2$
14: $X_2 \leftarrow X_2 + T_1$
```

```
Ensure: $P_2' \leftarrow P_2 + P_1$.
Ensure: $P_1' \leftarrow 2 \cdot P_1$.
    Point Addition
 1: $X_2 \leftarrow X_2 \cdot \boxed{\mathbf{Z_1}}$
 2: $Z_2 \leftarrow Z_2 \cdot X_1$
 3: $T_1 \leftarrow X_2 \cdot Z_2$
 4: $Z_2 \leftarrow Z_2 + X_2$
 5: $Z_2 \leftarrow Z_2 \cdot Z_2$
 6: $X_2 \leftarrow x \cdot Z_2$
 7: $X_2 \leftarrow X_2 + T_1$
    Point Doubling
 8: $X_1 \leftarrow X_1 \cdot X_1$
 9: $Z_1 \leftarrow Z_1 \cdot \boxed{\mathbf{Z_1}}$
10: $T_1 \leftarrow Z_1 \cdot \boxed{\mathbf{c}}$
11: $Z_1 \leftarrow Z_1 \cdot X_1$
12: $T_1 \leftarrow T_1 \cdot T_1$
13: $X_1 \leftarrow X_1 \cdot X_1$
14: $X_1 \leftarrow X_1 + T_1$
```

References

1. Aigner, H., Bock, H., Hütter, M., Wolkerstorfer, J.: A low-cost ECC coprocessor for smartcards. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 107–118. Springer, Heidelberg (2004)
2. Akishita, T., Takagi, T.: Power analysis to ECC using differential power between multiplication and squaring. In: Domingo-Ferrer, J., Posegga, J., Schreckling, D. (eds.) CARDIS 2006. LNCS, vol. 3928, pp. 151–164. Springer, Heidelberg (2006)
3. Amiel, F., Feix, B., Tunstall, M., Whelan, C., Marnane, W.P.: Distinguishing multiplications from squaring operations. In: Avanzi, R., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 346–360. Springer, Heidelberg (2009)
4. Batina, L., Mentens, N., Örs, S.B., Preneel, B.: Serial multiplier mrchitectures over GF(2^n) for elliptic curve cryptosystems. In: IEEE Mediterranean Electronical Conference - MELECON 2004, May 2004, pp. 779–782. IEEE (2004)
5. Batina, L., Mentens, N., Sakiyama, K., Preneel, B., Verbauwhede, I.: Low-cost elliptic curve cryptography for wireless sensor networks. In: Buttyán, L., Gligor, V., Westhoff, D. (eds.) ESAS 2006. LNCS, vol. 4357, pp. 6–17. Springer, Heidelberg (2006)
6. Bock, H., Braun, M., Dichtl, M., Hess, E., Heyszl, J., Kargl, W., Koroschetz, H., Meyer, B., Seuschek, H.: A milestone towards RFID products offering asymmetric authentication based on elliptic curve cryptography. Invited talk at RFIDsec 2008, July 2008
7. Brumley, D., Boneh, D.: Remote timing attacks are practical. Comput. Netw. **48**(5), 701–716 (2005)
8. Clavier, C., Feix, B., Gagnerot, G., Roussellet, M., Verneuil, V.: Horizontal correlation analysis on exponentiation. In: Soriano, M., Qing, S., López, J. (eds.) ICICS 2010. LNCS, vol. 6476, pp. 46–61. Springer, Heidelberg (2010)

9. Coron, J.-S.: Resistance against differential power analysis for elliptic curve cryptosystems. In: Koç, C.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 292–302. Springer, Heidelberg (1999)
10. Dhem, J.-F., Kœune, F., Leroux, P.-A., Mestré, P., Quisquater, J.-J., Willems, J.-L.: A practical implementation of the timing attack. In: Quisquater, J.-J., Schneier, B. (eds.) CARDIS 2000. LNCS, vol. 1820, pp. 167–182. Springer, Heidelberg (2000)
11. Eberle, H., Gura, N., Shantz, S.C., Gupta, V.: A cryptographic processor for arbitrary elliptic curves over GF(2^m). In: Deprettere, E., Bhattacharyya, S., Cavallaro, J., Darte, A., Thiele, L. (eds.) Application-Specific Systems, Architectures, and Processors - ASAP 2003, pp. 444–454, June 2003
12. Fürbass, F., Wolkerstorfer, J.: ECC processor with low die sizefor RFID applications. In: Proceedings of 2007 IEEE International Symposium on Circuits and Systems, May 2007. IEEE (2007)
13. Gebotys, C.H., Gebotys, R.J.: Secure elliptic curve implementations: an analysis of resistance to power-attacks in a DSP processor. In: Kaliski Jr, B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 114–128. Springer, Heidelberg (2003)
14. Großschädl, J.: A bit-serial unified multiplier architecture for finite fields GF(p) and GF(2^m). In: Koç, C.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 202–219. Springer, Heidelberg (2001)
15. Hankerson, D., Menezes, A.J., Vanstone, S.: Guide to elliptic curve cryptography. Springer, Heidelberg (2004)
16. Hartigan, J.A., Wong, M.A.: Algorithm AS 136: A K-Means Clustering Algorithm, vol. 28, pp. 100–108. Blackwell Publishing for the Royal Statistical Society, London (1979)
17. Herbst, C., Medwed, M.: Using templates to attack masked montgomery ladder implementations of modular exponentiation. In: Chung, K.-I., Sohn, K., Yung, M. (eds.) WISA 2008. LNCS, vol. 5379, pp. 1–13. Springer, Heidelberg (2009)
18. Homma, N., Miyamoto, A., Aoki, T., Satoh, A., Shamir, A.: Comparative power analysis of modular exponentiation algorithms. IEEE Trans. Comput. **59**(6), 795–807 (2010)
19. Kirschbaum, M., Popp, T.: Evaluation of power estimation methods based on logic simulations. In: Posch, K.C., Wolkerstorfer, J. (eds.) Proceedings of Austrochip 2007, 11 October 2007, Graz, Austria, pp. 45–51. Verlag der Technischen Universität Graz, Graz (2007). ISBN 978-3-902465-87-0
20. Kocher, P.C.: Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 104–113. Springer, Heidelberg (1996)
21. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 388–397. Springer, Heidelberg (1999)
22. Kumar, S.S., Paar, C.: Are standards compliant elliptic curve cryptosystems feasible on RFID? In: Workshop on RFID Security - RFIDSec 2006 (2006)
23. Lee, Y.K., Sakiyama, K., Batina, L., Verbauwhede, I.: Elliptic-curve-based security processor for RFID. IEEE Trans. Comput. **57**(11), 1514–1527 (2008)
24. López, J., Dahab, R.: Fast multiplication on elliptic curves over GF(2^{m}) without precomputation. In: Koç, C.K., Paar, C. (eds.) CHES 1999. LNCS, vol. 1717, pp. 316–327. Springer, Heidelberg (1999)
25. Mangard, S., Oswald, E.: Power analysis attacks - revealing the secrets of smart cards. Springer, Heidelberg (2007). ISBN 978-0-387-30857-9

26. Medwed, M., Oswald, E.: Template attacks on ECDSA. In: Chung, K.-I., Sohn, K., Yung, M. (eds.) WISA 2008. LNCS, vol. 5379, pp. 14–27. Springer, Heidelberg (2009)
27. Möller, B.: Securing elliptic curve point multiplication against side-channel attacks. In: Davida, G.I., Frankel, Y. (eds.) ISC 2001. LNCS, vol. 2200, pp. 324–334. Springer, Heidelberg (2001)
28. National Institute of Standards and Technology (NIST). FIPS-186-3: Digital Signature Standard (DSS). http://www.itl.nist.gov/fipspubs/ (2009)
29. NXP. Jcop 41 v2.3.1 java card (2007)
30. Orlando, G., Paar, C.: A high-performance reconfigurable elliptic curve processor for GF(2^m). In: Koç, C.K., Paar, C. (eds.) CHES 2000. LNCS, vol. 1965, pp. 41–56. Springer, Heidelberg (2000)
31. Örs, S.B., Oswald, E., Preneel, B.: Power-analysis attacks on an FPGA – first experimental results. In: Walter, C.D., Koç, C.K., Paar, C. (eds.) CHES 2003. LNCS, vol. 2779, pp. 35–50. Springer, Heidelberg (2003)
32. Oswald, E.: Enhancing simple power-analysis attacks on elliptic curve cryptosystems. In: Kaliski Jr, B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 82–97. Springer, Heidelberg (2003)
33. Öztürk, E., Sunar, B., Savas, E.: Low-power elliptic curve cryptography using scaled modular arithmetic. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 92–106. Springer, Heidelberg (2004)
34. Pan, W., Marnane, W.P.: A correlation power analysis attack against tate pairing on FPGA. In: Koch, A., Krishnamurthy, R., McAllister, J., Woods, R., El-Ghazawi, T. (eds.) ARC 2011. LNCS, vol. 6578, pp. 340–349. Springer, Heidelberg (2011)
35. Side-channel attack standard evaluation board. The SASEBO Website. http://staff.aist.go.jp/akashi.satoh/SASEBO/en/index.html
36. Walter, C.D.: Sliding windows succumbs to Big Mac attack. In: Koç, C.K., Naccache, D., Paar, C. (eds.) CHES 2001. LNCS, vol. 2162, pp. 286–299. Springer, Heidelberg (2001)
37. Walter, C.D.: Simple power analysis of unified code for ECC double and add. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 191–204. Springer, Heidelberg (2004)
38. Witteman, M.F., van Woudenberg, J.G.J., Menarini, F.: Defeating RSA multiply-always and message blinding countermeasures. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 77–88. Springer, Heidelberg (2011)
39. Wolkerstorfer, J.: Is elliptic-curve cryptography suitable for small devices? In: Workshop on RFID and Lightweight Crypto, 13–15 July 2005, Graz, Austria, pp. 78–91 (2005)

Implementations

Energy-Architecture Tuning for ECC-Based RFID Tags

Deepak Mane(✉) and Patrick Schaumont

Secure Embedded Systems, Center for Embedded Systems for Critical Applications, Bradley Department of ECE, Virginia Tech, Blacksburg, VA 24061, USA
{mdeepak, schaum}@vt.edu

Abstract. The implementation of Elliptic Curve Cryptography (ECC) on small microcontrollers is challenging. Past research has therefore emphasized performance optimization: pick a target architecture, and minimize the cycle count and footprint of the ECC software. This paper addresses a different aspect of resource-constrained ECC implementation: given the application profile, identify the most suitable architecture parameters. At the highest level, an application profile for ECC-based RFID tags is defined by the required security level, signature generation latency and the available energy/power budget. The target architecture parameters of interest include core-voltage, core-frequency, and/or the need for hardware acceleration. The paper brings two contributions to this complex design space exploration problem. First, we introduce a prototype setup for the precise energy measurement of a microcontroller-based ECC implementation. Second, we present a methodology to derive and optimize the architecture parameters starting from the application requirements. We demonstrate our methodology on a MSP430F5438A microcontroller, and present the energy/architecture design space for 80-bit and 128-bit security-levels, for prime field curves `secp160r1` and `nistp256`.

Keywords: Public key cryptography · Elliptic curves · RFID · Energy harvesting · Throughput · Digital signatures

1 Introduction

There are appealing advantages to using public-key cryptography (PKC) in RFID applications that require authentication. Indeed, PKC-based authentication significantly simplifies the distribution of cryptographic keys, resulting in a more scalable solution. The challenge of using ECC [8] in the RFID environment is how to deal with the high computational cost associated with ECC algorithms relative to the capabilities of the RFID platform [6,10,19,20]. Fortunately, this question has been reasonably well solved, and Table 1 shows some of the more recent achievements. It is fair to state that a security level of 80 bit (ECC curves of at least 160 bit) is within reach, with sub-second latency, in small footprint applications (i.e. 15 KGate, 100 KHz hardware implementations; or 16 bit, 8-MHz software implementations).

M. Hutter and J.-M. Schmidt (Eds.): RFIDsec 2013, LNCS 8262, pp. 147–160, 2013.
DOI: 10.1007/978-3-642-41332-2_10,

Table 1. Recent ECC implementations for RFID

Ref	Field/curve	Target Hardware or software	Time (s)	Operation	Resource
[11]	$GF(2^{163})$	HW, 0.13μm 100 Khz	0.244	Point mult	12,506 GE
[9]	$GF(2^{163})$	HW, 0.18μm 106 Khz	0.279	Point mult	11,904 GE
[12]	`secp160r1`	SW, MSP430F1611 8 MHz	8.54	ECDSA sign	13,520 bytes code
[21]	`secp160r1`	SW, MSP430F1611 8 MHz	1.1	Point mult	NA
[17]	`nistp192`	SW, MSP430F2131 6.7 MHz	1.6	Point mult	16,060 bytes code
[7]	`secp160r1`	SW, MSP430F5529 25 MHz	0.068	ECDSA sign	24,000 bytes code

To achieve these results, the authors of the designs in Table 1 need to use advanced algorithmic transformations and optimizations. Furthermore, they also make use of technology-specific features. Software implementations assume certain amounts of flash and RAM memory, or microcontroller features such as a hardware multiplier. Hardware implementations assume a target cell library of specific performance and feature size.

The assumption of a specific target architecture at the start of the design is typical for contemporary digital design methods. The designs in Table 1 are no exceptions. On the other hand, it is much harder for ECC designers to make clear commitments to application constraints such as the available energy budget and the required authentication latency. This is understandable: EDA tools do a poor job at estimating system performance and energy consumption. It is easier to optimize an implementation on a given target and then evaluate its characteristics on a prototype or using low-level simulation.

In this paper, we discuss the ECC RFID design problem by considering how to meet design requirements, rather than how to obtain the fastest ECC point multiplication. The main motivator for this is that the power source in the RFID environment is truly unique, and that this design problem deserves *a more holistic approach which considers energy source as well as energy consumer*.

Indeed, depending on the power source, RFID designs are either energy-constrained or else power constrained [4,5]. They also have to optimize application throughput (the time taken to complete a single signature) with respect to the available energy budget. We note that the requirements on energy and power depend on the type of RFID.

- Active RFID are powered from a battery source, and they have to minimize the energy consumed per signature as this will maximize the battery lifetime.
- Passive RFID are powered through an RF source, and they have to minimize the time required per signature while matching the available power budget.
- Passive RFID, powered through an energy harvesting mechanism that includes an energy store, have to minimize the energy used per signature as well, since this allows uninterrupted RFID operation. Furthermore, the energy needed for the desired application throughput has to match the average energy influx in the harvester.

The question addressed in this paper is: how can we select architecture parameters such that we meet these design requirements, including energy budget and application throughput? We provide an empirical answer to this question by presenting the energy/latency characteristics of an RFID doing ECDSA key generation, signature generation and signature verification. We present our results for a microcontroller target, a MSP430F5438A from Texas Instruments. We evaluate the energy characteristics of signatures at 80-bit and 128-bit security level [13]. We examine multiple architecture configurations: multiple frequencies, multiple core voltages, and with/without use of the MSP430's hardware multiplier. For each of these configurations, we carefully measure the required energy by tracking the MSP430 core current at high speed. The resulting curves then allow us to determine, for a given security level and energy budget, the most appropriate core frequency and voltage level.

In a nutshell, our results are as follows. Increasing the MSP430 core frequency always reduces the energy consumed per signature. This is because energy consumed by leakage (i.e. static power dissipation) becomes proportionally less important as the runtime of the application decreases. Hence, in a given energy harvesting method, it is always better to wait as long as possible before initiating ECC computations and, once sufficient energy is available, complete them as quickly as possible at the highest possible operating frequency. A second observation is that the impact of security level on energy budget is significant. For an MSP430 without a hardware multiplier, our prototype needs roughly six times as much energy per signature at the 128 bit security level compared to the 80 bit security level. When a hardware multiplier can be used, the difference is roughly two times. A third observation is that architecture specialization matters. Under constant security level, a hardware multiplier reduces the energy consumption by almost 8 times. Voltage scaling results in an additional gain factor of 2 in energy.

The remainder of the paper is organized as follows. In the next section, we describe the background related to power and energy in the digital electronics. In Sect. 3, we review the target ECC design measured using our method. Next, we introduce the target platform for our experiments. Section 5 describes a setup that can be used for precise energy measurement of RFID applications. The resulting energy/throughput curves are presented in Sect. 6, and applied in a methodology in Sect. 7. Section 8 concludes the paper.

2 Background: Power and Energy in Digital Electronics

Power dissipation in modern digital electronics has two major components: static power dissipation defined by static leakage current, and dynamic power consumption, defined by circuit activity. The static power dissipation depends, in first order, on the operating voltage of the circuit and the size of the circuit. Dynamic power dissipation depends, in first order, on the operating frequency of the circuit, the size of the circuit, and the square of the operating voltage.

We analyze what happens to the energy dissipation for a fixed workload, such as signature verification, under varying operating conditions. In the following, K

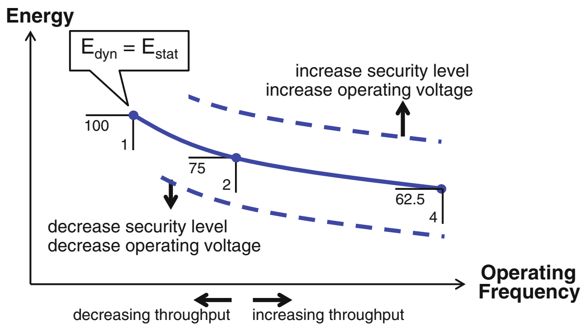


Fig. 1. Expected energy dissipation as a function of frequency

and C are technology constants, α is an application-dependent activity factor, T_{cycle} is the clock cycle period, f_{cycle} is the operating frequency, and n is the workload cycle budget. The energy dissipation per workload has a static and a dynamic component.

$$E_{dyn} = P_{dyn}.T_{alg} = \alpha.C.V^2.f_{cycle}.T_{cycle}.n \tag{1}$$

$$E_{stat} = P_{static}.T_{alg} = K.V.T_{cycle}.n \tag{2}$$

The total energy dissipation thus equals

$$E_{tot} = E_{dyn} + E_{static} = n.[\alpha.C.V^2 + K.V.T_{cycle}] \tag{3}$$

This formulation leads to the following assessment. If the clock frequency increases, the total energy per workload will decrease: the dynamic energy remains constant, while the static energy decreases. Furthermore, if the operating voltage decreases, the total energy per workload will decrease as well. Finally, if the security level of the design increases (from 80 to 128 bit, for example), the cycle budget n will increase, and the total energy per workload will increase as well. This analysis is captured by Fig. 1. The leftmost point on the curve represents a design where the static and dynamic parts of the energy per workload are equal. As the operating frequency increases, the total energy decreases as well. We note that this figure is a theoretical model: it ignores overhead for clock generation, and for transitions between power modes.

3 Target Protocol: ECDSA in `secp160r1` and `nistp256`

The driving application for our measurements is ECDSA key generation, signature generation, and signature verification. Rather than developing our own implementation from scratch, we use the RELIC library with support for the MSP430 and 32-bit hardware multiplier [16]. We implement two prime-field curves, `secp160r1` and `nistp256`. The scalar multiplication is done with a left-to-right window-3 NAF multiplication, and using Jacobian Projective Coordinates. The field operations are basic Comba multiplication and squaring, with Montgomery reduction. SHA-1 is used for hashing and as a pseudorandom generator. We used two implementation variants for each of the curves: one which uses a 32-bit hardware multiplier (using RELIC's `msp-asm` backend), and a second one which emulates multiplication in software (using RELIC's `easy` backend).

Our standard testbench goes through ECDSA key generation, ECDSA signature generation of a fixed digest, and ECDSA signature verification. The execution time, as well as the energy, is measured for each of these steps separately. Our performance and energy numbers only cover the computations, and they don't include initialization overhead, or overhead from data communications.

4 Target Platform: MSP430F5438A

We use MSP430F5438A [1] as our prototyping platform. The MSP430F5438A is an ultra-low power Reduced Instruction Set Computer (RISC) from Texas Instruments, optimized for low-resource applications [2]. The architecture combines five different low power modes suitable for low power battery operation. The MSP430F5438A features a 16-bit CPU, 256 KB flash, 16 KB SRAM, up to 25 MHz CPU clock and 16 working registers with 12 available as general purpose registers. It also supports a 32 bit hardware multiplier.

Figure 2 shows the internal energy management architecture of the MSP430F5438A. In general, `VCore` supplies the CPU, memories (flash and RAM), and the digital modules, while `DVcc` supplies the I/Os and all analog modules. The internal core voltage of the MSP430F5438A, `VCore`, needs to be adjusted as a function of the desired operating frequency. The `VCore` output is programmable in four steps, to provide only as much power as is needed for the speed that has been selected for the CPU. We configure `VCore` voltage by writing register bits `PMMCOREV[1:0]`. Table 2 shows recommended `PMMCOREV` settings and minimum external power supply voltage for different frequency ranges. We make use of these programmable `VCore` levels to optimize the energy efficiency of ECDSA on the MSP430.

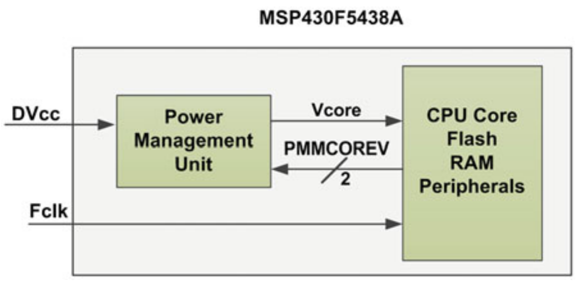


Fig. 2. Voltage and frequency scaling

Table 2. Recommended PMMCOREV and DV_{CC} settings for selected f_{sys}

f_{sys}(max) (MHz)	Minimum DVCC (V)	Minimum PMMCOREV[1:0]
8	1.8	00
12	2.0	01
20	2.2	10
25	2.4	11

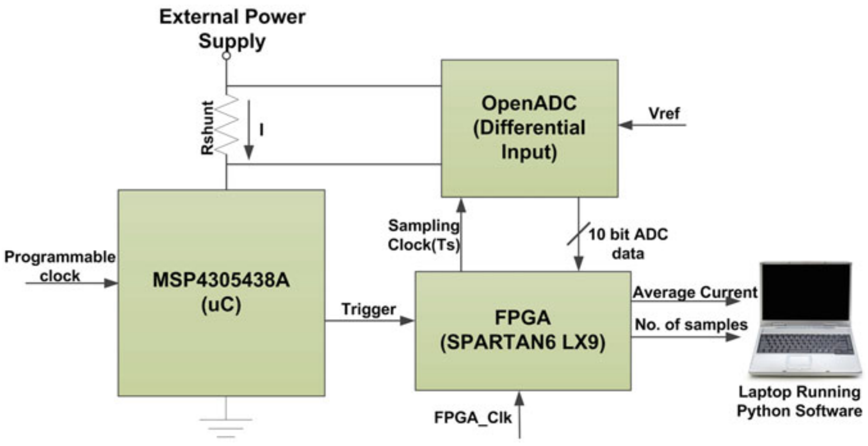


Fig. 3. Energy measurement setup block diagram

5 High Resolution Energy Measurement

Figure 3 depicts the block diagram of energy measurement setup used in our experiments. The average current consumed by a microcontroller during a particular interval of time is measured by integrating the immediate current. The current is measured by means of the voltage drop over a shunt resistor on the microcontroller `Vcc` line. To sample the voltage drop, we use OpenADC [14,15] with a Spartan FPGA [3], in place of a traditional high-speed oscilloscope setup. OpenADC is a custom ADC board with a 10-bit A/D converter that supports differential inputs and an adjustable reference voltage. The FPGA takes care of sample accumulation and sample counting.

The integration period is defined by means of trigger signals created by the microcontroller. This requires a simple instrumentation of the software application. The clock frequency of the A/D converter can be independently chosen of the microcontroller clock.

Procedure to measure average current: At a desired time, the microcontroller triggers the FPGA to start sampling OpenADC data (Fig. 4). Once trigger is asserted, FPGA accumulates ADC samples until trigger line is re-asserted. The accumulator represents the average current consumed by a function being executed on a microcontroller. It also measures number of samples collected during

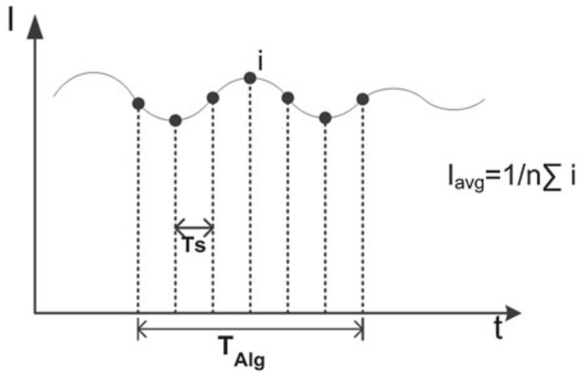


Fig. 4. Average current measurement

trigger window to derive the execution time of that function. The FPGA design then provides average current and number of samples collected to a python script running on laptop (PC). A Python script uses this data and calculates the average energy consumed by an MSP430 function as follows.

$$\begin{aligned}\text{Energy} &= V_{cc}.I_{avg}.T_{Alg}\\ &= V_{cc}.\frac{Accm.V_{ref}}{2^n.N_s.R}.N_s.T_s\\ &= V_{cc}.\frac{Accm.V_{ref}}{2^n.R}.T_s\end{aligned} \tag{4}$$

$$\text{Cycle count} = N_s.T_s.F_{cpu} \tag{5}$$

where V_{cc} is supply voltage of the microcontroller, $Accm$ is FPGA accumulator value, V_{ref} is ADC reference voltage, N_s is number of samples collected during trigger window, T_s is sampling period, 2^n is resolution of ADC where n is 10 bit, R is a shunt resistor value and F_{cpu} is microcontroller frequency. Our Energy formula assumes that the voltage supply at the microcontroller input is constant, in other words, that the voltage drop over the shunt resistor is negligible with respect to V_{cc}.

The above formula shows that the resolution of energy measurements increases with low reference voltage of ADC, high sampling frequency and with high resolution of ADC. We use sampling rate of 20 MHz for operating frequencies below 15 MHz and 30 MHz for operating frequencies above 15 MHz. In our experiments, ADC reference voltage is 0.5 V and shunt resistor is 100 Ω. This setup can be used to measure the energy consumption of any device provided that it has the facility to insert a resistor in series with power supply. Also, the device should be able to generate a trigger signal to activate the FPGA accumulator.

Table 3. Cycle count of ECDSA operations on the MSP430F5438A

Operation	`secp160r1`		`nistp256`	
	w/o hardware multiplier	with hardware multiplier	w/o hardware multiplier	with hardware multiplier
Key generation	19,343,970	1,796,499	124,427,469	5,225,820
Signing	19,141,737	2,372,103	124,942,312	6,408,792
Verification	57,621,281	57,48,345	346,290,644	15,584,937

Table 4. Code size of the implementation of ECDSA on the MSP430F5438A

	`secp160r1`		`nistp256`	
	w/o hardware multiplier	with hardware multiplier	w/o hardware multiplier	with hardware multiplier
Flash bytes	27,134	28,168	28,138	32,234
RAM bytes	1,074	1,074	1,542	1,542

6 Results

In this section, we present experimental results of our energy measurements under different architecture configurations. We measure the energy consumption of ECDSA key generation, signature generation and signature verification for different operating voltages and frequency settings. We also measure the runtime for each operation in order to obtain the throughput. We use the `gcc 4.6.3` cross-compiler for MSP430 family of microcontrollers. Table 3 shows cycle counts for different ECDSA operations, and Table 4 shows the footprint of the implementations.

The graphs show the energy/throughput characteristics for signature generation.

Figures 5 and 6 show energy consumption for 80-bit(`secp160r1` curve) security level, without and with hardware multiplier, respectively. We analyze the effect of different operating voltage on energy consumption. In our experiments, we found that if the microcontroller is operated at 2.0 V instead of 2.7 V, it saves almost 1.4 times energy consumption. Reducing operating voltage reduces dynamic power consumption because the circuit will have smaller voltage swings during switching. Also the static power consumption reduces because the leakage current reduces. Further, the use of the hardware multiplier results in almost 8 times energy reduction. This is because the hardware multiplier accelerates the signing operation almost by 8 times.

Figures 7 and 8 show energy consumption for 128-bit(`nistp256` curve) security level, without and with hardware multiplier respectively.

The curves for operation at 2.7 V is discontinuous at 12 MHz and 20 MHz. The discontinuities are caused by reprogramming of the power management system of the CPU.

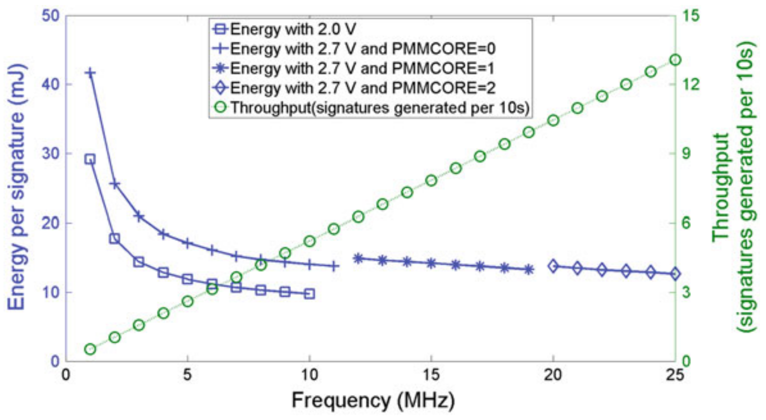


Fig. 5. Energy consumption for `secp160r1` without hardware multiplier

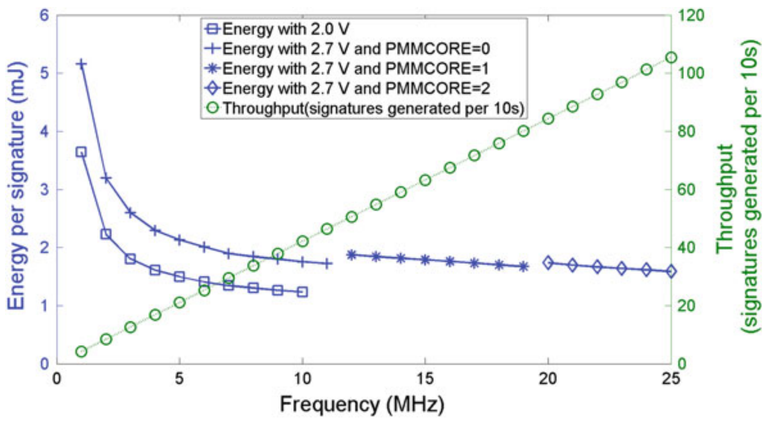


Fig. 6. Energy consumption for `secp160r1` with hardware multiplier

Figure 9 shows energy improvement factors for different architecture configurations. It can be observed that moving towards the origin results in the most optimized design where energy consumption is minimal. Reducing the operating supply voltage from 2.7 to 2.0 V results in a gain of 1.4 for both security levels. The computational complexity of used algorithm results in different improvement factors since energy consumption is dependent upon runtime of an algorithm. The acceleration achieved with hardware multiplier is also dependent on the used security level which gives different energy gain factors. Overall the most significant impact comes from architecture specialization.

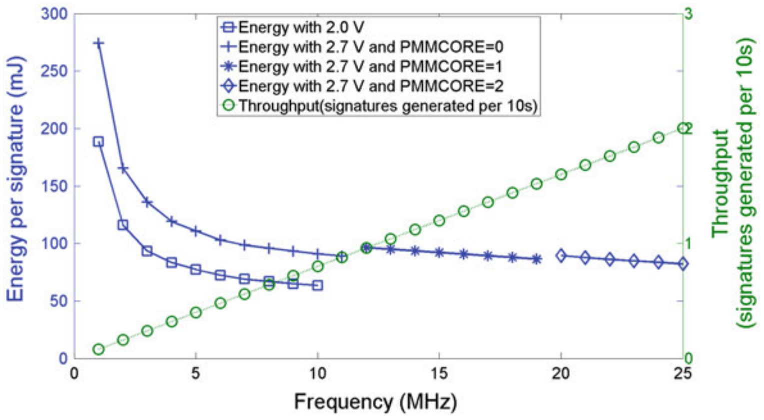


Fig. 7. Energy consumption for `nistp256` without hardware multiplier

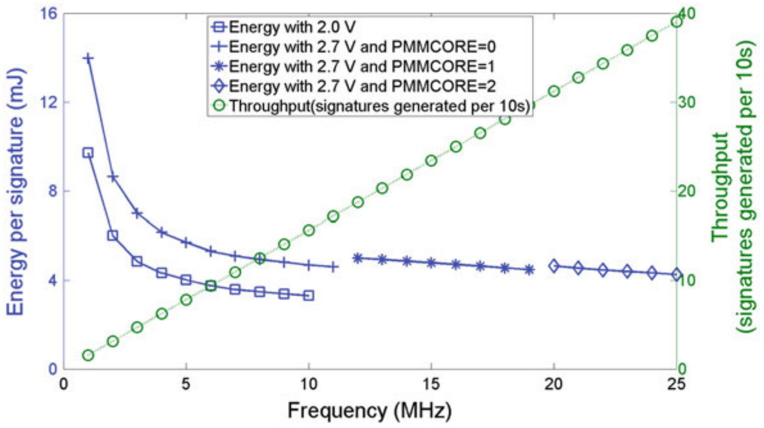


Fig. 8. Energy consumption for `nistp256` with hardware multiplier

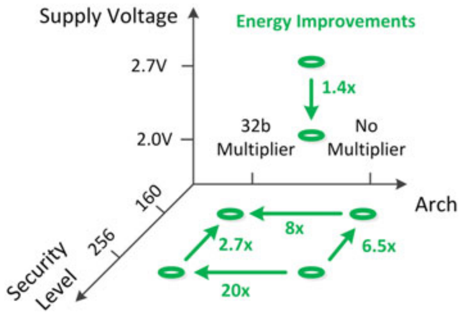


Fig. 9. Energy Profile for ECDSA `secp160r1` and `nistp256` on TI `MSP430F5438A`

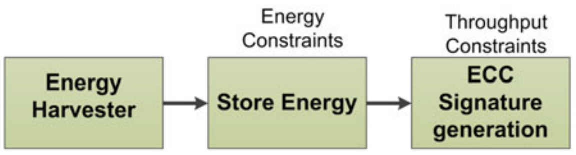


Fig. 10. Energy harvesting system

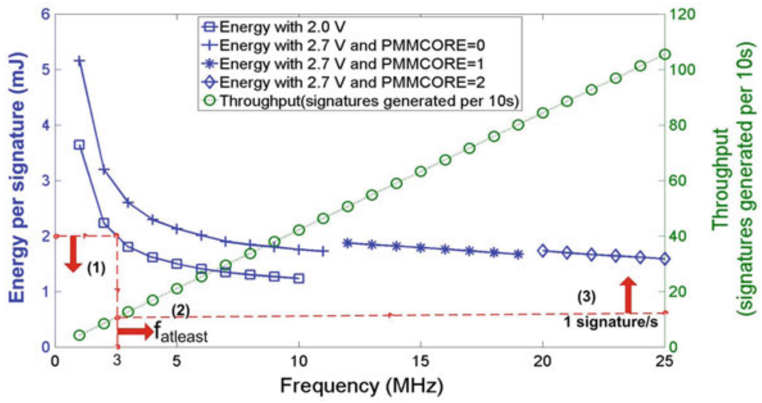


Fig. 11. Architecture tuning for energy constrained system

7 Methodology for Architecture-Energy Tuning

Finally, we show how the energy measurement method can be applied to meet the design requirements. Figure 10 shows a typical energy harvesting [18] setup where energy is harvested and stored. When sufficient energy is available, the application can execute. In the following examples, we demonstrate how our energy throughput curves can be used for system dimensioning. We consider two cases. In a first case, start from an energy constraint and derive the achievable performance, expressed as the latency to complete one signature. This case is relevant when we use an energy store of a given dimension, and we would like to evaluate what system performance can be achieved. In the second case, we start from a desired application performance, and we derive the required energy (and thus the required energy store). Since there are multiple energy/throughput curves (at different security levels, and at different architecture configurations), we propose that this evaluation is done concurrently over all curves available, in order to analyze the design space. In the example discussed below, however, we will focus on a single security level and a microcontroller with a hardware multiplier.

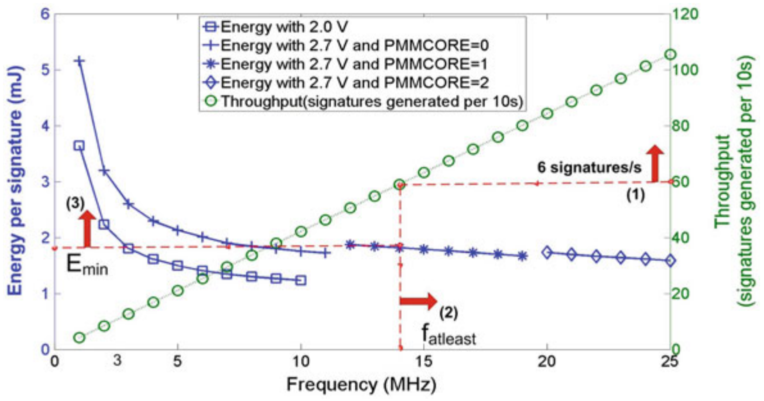


Fig. 12. Architecture tuning for throughput constrained system

Figure 11 shows the energy curve for the 80-bit security level. We assume an energy limit of 2 mJ per signature, and we assume a 2 V operating level. This limit sets a *minimum* operating frequency for the microcontroller. The 2 mJ energy level requires a 3 MHz operating frequency, which enables a signature to complete in a one second. If we increase the core voltage to 2.7 V, the minimum operating frequency at 2 mJ per signature will increase to 9 MHz, and the signature will be done in 0.25 s.

A second example is to start from the required signature throughput. The example shown in Fig. 12 needs to complete a signature in 1/6 of a second. We use our graphs to decide the operating frequency and to find required energy per signature. First, we note that the system needs to operate at least at 14 MHz. At that frequency, only the 2.7 V core voltage mode is available. Under this setting, the corresponding energy per signature is 1.9 mJ.

8 Conclusion

We demonstrated the importance of energy-architecture tuning for RFID. The complex energy-provisioning environment of these applications requires a holistic approach that not only considers performance optimization, but also the energy and/or power needs. We analyzed and quantified, for two different security levels, the impact of several architecture optimization techniques including voltage scaling, frequency scaling, and the use of a hardware multiplier. Using the analysis presented in this paper, we can now investigate the design of a secure RFID with integrated energy harvesting.

Acknowledgements. This project was supported in part through NIST, Grant no. 60NANB10D004.

References

1. Texas Instruments: MSP430F5438A Mixed Signal Microcontroller. http://www.ti.com/lit/ds/symlink/msp430f5438a.pdf
2. Texas Instruments: MSP430x5xx and MSP430x6xx Family User's Guide (2013). http://www.ti.com/lit/ug/slau208m/slau208m.pdf
3. Xilinx Spartan-6 FPGA LX9 MicroBoard. http://www.em.avnet.com/en-us/design/drc/Pages/Xilinx-Spartan-6-FPGA-LX9-MicroBoard.aspx
4. Buettner, M., Greenstein, B., Wetherall., D.: Dewdrop: An Energy-Aware Runtime for Computational RFID. In: Proceedings of USENIX Symposium on Networked Systems Design and Implementation (NSDI), Boston, MA, USA (2011)
5. Buettner, M., Greenstein, B., Wetherall, D.: Dewdrop: An energy-aware runtime for computational rfid. In: Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation (NSDI 2011), p. 15. USENIX Association, Berkeley (2011). http://dl.acm.org/citation.cfm?id=1972457.1972478
6. Chae, H.J., Yeager, D.J., Smith, J.R., Fu, K.: Maximalist cryptography and computation on the WISP UHF RFID tag. In: Proceedings of the Conference on RFID Security, July 2007. http://prisms.cs.umass.edu/kevinfu/papers/chae-RFIDSec07.pdf
7. Gouvêa, C.P.L., López, J.: High speed implementation of authenticated encryption for the MSP430X microcontroller. In: Hevia, A., Neven, G. (eds.) LATINCRYPT 2012. LNCS, vol. 7533, pp. 288–304. Springer, Heidelberg (2012)
8. Hankerson, D., Menezes, A.J., Vanstone, S.: Guide to Elliptic Curve Cryptography. Springer, New York (2003)
9. Hein, D., Wolkerstorfer, J., Felber, N.: ECC is ready for RFID – A proof in silicon. In: Avanzi, R.M., Keliher, L., Sica, F. (eds.) SAC 2008. LNCS, vol. 5381, pp. 401–413. Springer, Heidelberg (2009)
10. Hinterwälder, G., Paar, C., Burleson, W.P.: Privacy preserving payments on computational RFID devices with application in intelligent transportation systems. In: Hoepman, J.-H., Verbauwhede, I. (eds.) RFIDSec 2012. LNCS, vol. 7739, pp. 109–122. Springer, Heidelberg (2013)
11. Lee, Y.K., Sakiyama, K., Batina, L., Verbauwhede, I.: Elliptic-curve-based security processor for rfid. IEEE Trans. Comput. **57**(11), 1514–1527 (2008)
12. Liu, A., Ning, P.: Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks. In: Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN 2008), pp. 245–256. IEEE Computer Society, Washington (2008). http://dx.doi.org/10.1109/IPSN.2008.47
13. National Institute of Standards and Technology: FIPS 186–3: Digital Signature Standard (DSS) (2009). http://www.itl.nist.gov
14. O'Flynn, C.: OPENADC (2012). http://newae.com/tiki-index.php?page=OpenADC
15. O'Flynn, C.: Power analysis for cheapskates (2012). https://media.blackhat.com/ad-12/O%27Flynn/bh-ad-12-for-cheapskates-o%27flynn-WP.pdf
16. Oliveira, L.B., Aranha, D.F., Gouvêa, C.P.L., Scott, M., Câmara, D.F., López, J., Dahab, R.: Tinypbc: pairings for authenticated identity-based non-interactive key distribution in sensor networks. Comput. Commun. **34**(3), 485–493 (2011)
17. Pendl, C., Pelnar, M., Hutter, M.: Elliptic curve cryptography on the WISP UHF RFID tag. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 32–47. Springer, Heidelberg (2012)
18. Raju, M.: T. Instruments White Paper: Energy Harvesting (2008)

19. Ransford, B., Clark, S., Salajegheh, M., Fu, K.: Getting things done on computational rfids with energy-aware checkpointing and voltage-aware scheduling. In: Proceedings of the 2008 Conference on Power Aware Computing and Systems (HotPower 2008), p. 5. USENIX Association, Berkeley (2008). http://dl.acm.org/citation.cfm?id=1855610.1855615
20. Wang, H., Li, Q.: Efficient implementation of public key cryptosystems on mote sensors (short paper). In: Ning, P., Qing, S., Li, N. (eds.) ICICS 2006. LNCS, vol. 4307, pp. 519–528. Springer, Heidelberg (2006). http://dx.doi.org/10.1007/11935308_37
21. Wenger, E., Werner, M.: Evaluating 16-bit processors for elliptic curve cryptography. In: Prouff, E. (ed.) CARDIS 2011. LNCS, vol. 7079, pp. 166–181. Springer, Heidelberg (2011)

Speed and Size-Optimized Implementations of the PRESENT Cipher for Tiny AVR Devices

Konstantinos Papagiannopoulos and Aram Verstegen(✉)

Radboud Universiteit Nijmegen, Nijmegen, The Netherlands
aram.verstegen@gmail.com

Abstract. This paper presents high-speed and low-size assembly implementations of the 80-bit version of the PRESENT cipher for the (Tiny)AVR family of microcontrollers. We report new speed and size records for our implementations, with the speed-optimized version achieving a full encryption in 8721 clock cycles and the size-optimized version compressing the cipher down to 272 bytes; the previous state of the art for (Tiny)AVR achieved 10723 clock cycles for encryption with a size of 936 bytes. Along with the two implementation extrema (speed and size optimized versions), we offer insight into techniques and representations that show the speed/area tradeoffs and provide intermediate solutions for various configurations.

Keywords: PRESENT · AVR · ATtiny · Assembly software implementation

1 Introduction

Modern society is constantly witnessing an extensive and large scale deployment of tiny computing devices. Information processing and wireless communication are being thoroughly integrated into everyday objects and activities, developing a large distributed mobile infrastructure and ushering in the era of ubiquitous computing. RFID tags attached to products, cardiac pacemakers, fire-detecting sensor nodes and the like need to operate *securely* under particularly *restricted conditions*, namely low battery life, small processing power and bandwidth-demanding ad-hoc network protocols. To achieve sustainable security in this new landscape, researchers have developed new cryptographic primitives and techniques, namely lightweight cryptographic ciphers such as PRESENT [5], Klein [11], LED [13] and others.

The majority of these ciphers was designed with hardware performance in mind, leaving most software implementations relatively inefficient. For instance, the AVR-Crypto-Lib [21] often resorts to C language implementations, resulting in 100.000 clock cycles for a single encryption with PRESENT. AVR microcontrollers are often encountered regarding the Internet of Things and ubiquitous computing, thus they are an interesting platform on which to enable and optimize lightweight ciphers. The University of Louvain has initiated a project to draft efficient assembly implementations of various lightweight ciphers on

M. Hutter and J.-M. Schmidt (Eds.): RFIDsec 2013, LNCS 8262, pp. 161–175, 2013.
DOI: 10.1007/978-3-642-41332-2_11,

resource-constrained AVR devices to make fair comparisons about their relative efficiency. Resulting from this project is the state of the art in both speed and size: an implementation [8] which achieves an encryption with PRESENT in 10723 clock cycles in 936 bytes.

Our contribution. This paper describes the details of a speed optimized [20] and a size-optimized [22] implementation of the PRESENT cipher on the ATtiny45, attained with the aid of algorithmic improvements and efficient programming techniques. Our speed-optimized version improves the state of the art [8] by an 18 % reduction in clock cycles, while the size-optimized version is 70 % smaller.

Algorithmic improvement:

– A merged SP layer, i.e. combining the substitution and permutation layer of the cipher in order to construct lookup tables that remove the time-consuming permutations [10, 12]. This optimization constitutes the core of the achieved speed improvement.

Programming improvements:

– Squared S-Box representations, i.e. S-Boxes which are custom-made for fast access in the AVR 8-bit architecture, also used by Eisenbarth [8].
– Compact S-Box representations, i.e. minimal footprint S-Boxes that reduce the implementation size.
– Minimal key register rotations, allowing the key update procedure to complete in fewer instructions.
– Memory access optimizations, grouping memory transactions to improve speed.
– Algorithm serialization, by keeping part of the state in SRAM while we operate on fewer registers.
– Indirect register access to let loops drive repeated operations on CPU registers.
– Use of the stack to store intermediate values to avoid using more dedicated registers.
– Code restructuring and efficient procedure callings.

2 Background

2.1 PRESENT Cipher

PRESENT [5] is an ultra-lightweight, 64-bit symmetric block cipher, using 80-bit or 128-bit keys. It is based on a substitution/permutation network and it is named as a reference to Serpent [2] due to its similar constructs. As of 2012, PRESENT (among other ciphers) was adopted by ISO as a standard for a lightweight block cipher (ISO/IEC 29192-2:2012 [17]). The full algorithm has so far been resistant to attempts at cryptanalysis, although the most successful attack has shown that up to 15 of its 31 rounds can be broken with $2^{35.6}$ plaintext-ciphertext pairs in 2^{20} operations [1, 7, 19].

PRESENT uses exclusive-or as its round key operation, a 4-bit substitution layer, a 4-bit period bit position permutation network in 31 rounds and a final round key operation. Key scheduling is a combination of bit rotation, S-Box application and exclusive-or with the round counter. Constructs found in PRESENT are also encountered in SPONGENT [4], in hash function constructs based on block ciphers as proposed by Hirose [6,14,15] (H-PRESENT) and in the similar Maya [9] or generalized SMALLPRESENT [18]. Thus the optimizations presented here can also be of interest with respect to the implementations of these ciphers. In our approach, we have implemented PRESENT for the recommended 80-bit key size in AVR assembly in two versions, optimized for maximal speed and minimal size. Support for 128-bit keys was also added to the size-optimized implementation as the required extra registers became available through optimizations, but we will focus on the implementation of the variant with 80-bit keys.

2.2 PRESENT Algorithm

The cipher's key register is supplied with the 80-bit cipher key and in every encryption round the first 64 bits of the 80-bit key register form the round key. To encrypt a single 64-bit block, during each encryption round, PRESENT applies an exclusive-or (XOR) with the current round key followed by a substitution and a permutation layer. The substitution layer applies nibble-wise (4-bit) S-Boxes to the state, while the permutation layer re-arranges the bits in the state following a 4-bit period. Key scheduling is done by rotating the key register 61 bit positions to the left, applying the S-Box to the top nibble of the key register and XORing bits 15 through 19 with the round counter. There is a total of 31 such rounds and finally the round key is applied one last time (Fig. 1).

2.3 The 8-bit Family of TinyAVR Microcontrollers

Atmel offers a wide range of 8-bit microcontrollers, including high performance devices (ATxmega), mid-range devices (ATmega) and low-end devices with limited memory, storage and processing power (ATtiny). Common applications of AVR microcontrollers include smart cards, motor control systems, medical applications et cetera.

Our focus is on the resource-constrained ATtiny architecture, which typically features less than 1 KB of static RAM (SRAM) and flash storage ranging from 1 to 8 KB. The architecture uses 32 general-purpose registers, *R0-R31*.

Several registers have special characteristics, namely registers *R0* through *R15* can not be accessed by instructions that provide an immediate value as an operand, i.e. you can only perform memory access from these registers. In addition, register pairs *R27:R26* (denoted as X), *R29:R28* (denoted as Y) , and *R31:R30* (denoted as Z) can access the SRAM. The Y register can also be used for indirect register addressing, the Z register can also access the flash storage and be used for indirect jumps and calls. Instructions using these 3

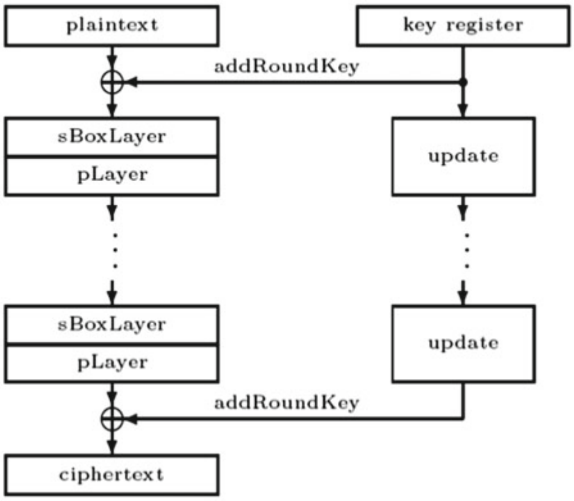


Fig. 1. Schematic of the PRESENT cipher. It consists of 31 rounds, including XOR round key application, nibble-wise substitution, bit position permutation and key update.

pointer register pairs also allow post-increment and pre-decrement of the pointer. At all times, the 6 special registers can be utilized as general-purpose registers.

The ATtiny instruction set consists of the basic 90 single-word instructions found in all AVR architectures. However, due to the limited size of its core, it does not support the extended instruction set which includes multiplication, in contrast with the ATmega architecture.

ATtiny configuration. We perform all simulations on the ATtiny45, which has a maximum clock frequency of 20 MHz, 256 bytes of SRAM and 4 KB of flash storage.

Radix-2^8 representation. The PRESENT cipher requires representing integers of size 64 and 80 bits. Thus, we split the number into 8-bit components, using radix-2^8 and an m-bit integer is represented as $n = \lceil m/8 \rceil$ bytes $(x_0, x_1, \ldots, x_{n-1})$ such that $x = \sum_{i=0}^{n} x_i 2^{8*i}$.

3 High-Speed Implementation

3.1 PRESENT S-Box and P-Layer Implementation

In this section we examine the S-Box of the PRESENT cipher from the speed perspective, using lookup tables, and we offer several variations, utilizing the speed-area trade-off. We analyze the *original S-Box*, identify its performance issues and suggest two possible lookup table representations (with 8 and 16 bytes respectively). Subsequently, we expand it to the faster, yet space-demanding

Table 1. The original S-Box of the PRESENT cipher.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S[x]	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Table 2. The packed representation of the original S-Box, using 8 bytes. Each table column represents two substitutions. This would give a size optimization of 8 bytes to begin with, but considerations for unpacking code apply. (See Sect. 4.2.)

x	01	23	45	67	89	AB	CD	EF
S[x]	C5	6B	90	AD	3E	F8	47	12

(256 bytes) *Squared S-Box*. In the last section, we examine the *combination of the S-Box and the permutation layer*, resulting in a very large lookup table (1024 bytes) that substantially boosts performance.

Original S-Box. The original S-Box, presented by Bogdanov et al. [5] consists of 16 different substitutions, each with a 4-bit input and a 4-bit output, as shown in Table 1.

Representation. If we aim for a particularly small size footprint, it is possible to use either a *packed* or an *unpacked* representation of the original S-Box. The original, *unpacked* version stores the lookup table in 16 bytes, where every 4-bit input to 4-bit output substitution is stored using 8 bits of space (i.e. there exists redundancy of in the representation). The *packed* version, stores *two* 4-bit input to 4-bit output substitutions using 8 bits, i.e. without any redundancy, resulting in an 8 byte lookup table.

Performance. The core performance issue regarding the 4-bit S-Box is the penalty in accessing it, if stored in a lookup table. The AVR architecture is designed to enable fast access for 8-bits at a time. Thus, a lookup table of the original S-Box is rather small (16 bytes for an unpacked version, 8 bytes for a packed one), but we can assume it is also relatively inefficient speed-wise due to the overhead operations that need to take place before and after each table lookup. Surely the packed S-Box (Table 2) is the least speed-efficient variant, since after the 4-bit lookup we also have to extract the upper or lower half. This issue is not encountered in the unpacked version, which makes better use of the AVR 8-bit architecture. However, performance is not optimal due to the fact that we only substitute 4 bits at a time, while we use 8-bit operations, i.e. the redundant representation results in more memory accesses than needed.

Squared S-Box. A solution to the aforementioned performance problem is to construct a new lookup table that: (a) is custom made for the 8-bit AVR architecture, like the unpacked S-Box and (b) uses a non-redundant representation similar to the packed S-Box.

Representation. In Table 3, we demonstrate a *Squared S-Box*, which uses an 8-bit input and produces an 8-bit output. Within an 8-bit space, we can contain two 4-bit substitution values and the number of possible substitution values is

Table 3. The 256-byte Squared S-Box. It substitutes one byte at a time, without any overhead or redundancy.

x	00	01	02	03	...	0C	0D	0E	0F
S[x]	CC	C5	C6	CB	...	C4	C7	C1	C2
x	10	11	12	13	...	1C	1D	1E	1F
S[x]	5C	55	56	5B	...	54	57	51	52
⋮	⋮	⋮	⋮	⋮	...	⋮	⋮	⋮	⋮
x	F0	F1	F2	F3	...	FC	FD	FE	FF
S[x]	2C	25	26	2B	...	24	27	21	22

16, thus the total size of the lookup table is $16 \cdot 16 = 256$ bytes. As a result, there is no need for overhead computation and the substitution consists of a single lookup. This approach has also been followed before by Eisenbarth [8].

Performance. The Squared S-Box described is an efficient and viable solution with respect to the cipher's substitution layer. It is custom made for the 8-bit AVR architecture and allows us to perform byte substitutions with a single flash memory lookup. Furthermore, it is relatively size-efficient, consisting of 256 bytes. We consider it could almost be transfered to ATtiny45 SRAM from flash memory during the initialization process of the algorithm, but we would be left without any stack space and therefore it would certainly be a special purpose implementation. That memory transfer *is* viable for applications that do dedicated encryption or decryption (provided we are left with some stack space for the workings of the rest of the algorithm). The instruction to load from flash (`lpm`) takes 3 clock cycles, whereas loading from or saving to SRAM (`ld`) takes only 2. Given the fact that the block size of the PRESENT cipher is 64 bits, it requires $64/8 = 8$ S-Box lookups per round. The cipher consists of 31 rounds so a full encryption requires $8 \cdot 31 = 248$ S-Box applications. If an application would singularly encrypt or decrypt, we could save 248 cycles per encryption after an initial $(3 + 2) \cdot 256 = 1280$ clock cycles. That means that the processing penalty to load a 256 byte table from flash storage to SRAM would not be compensated until over 5 encryptions or decryptions were computed sequentially, so this approach is not feasible for the general case of intermixed encryption and decryption.

Merged SP Lookup Table. Although the Squared S-Box lookup table solution is viable for the PRESENT cipher, we need a speed optimization for the most complex part of the PRESENT algorithm: the permutation layer. In contrast to hardware implementations, where it is a trivial rewiring of outputs, most software implementations require a large number of bit rotation and move operations. The AVR architecture is not capable of performing fast shifts and rotations (compared to for instance the ARM11 processors, which can do shifts and rotations for free), i.e. an n-bit shift requires n clock cycles, and thusly we have to look for alternatives. In this direction, work by Hutter and Schwabe [16]

```
mov ZH, high     ; load high part of Z address
mov ZL, low      ; load low part of Z address
lpm register , Z ; load from Z addresses into register
```

Fig. 2. Flash memory is addressed through 16-bit pointers. We aim to keep changes in the high byte to a minimum.

on ATmega[1] suggested the usage of multiplications instead of rotations/shifts, however this is not viable on ATtiny chips due to the fact that they do not possess native multiplication instructions.

Representation. The fastest approach that we identified for the permutation layer is the idea developed by Zheng Gong and Bo Zhu [10,12]. Due to the adaptation of this novel approach in the AVR architecture, we are able to improve performance over the state of the art [8]. Specifically, Gong and Zhu exploited the internal structure of the permutation layer, i.e. the fact that every output of a 4-bit S-Box will contribute one bit to the cipher. The underlying pattern for the permutation is the following:

$$for\ k : old\ position,\ l : new\ position,\ l = f(k) = 16 \cdot (k \bmod 4) + (k \div 4)$$

Thus, the first 2 bits of the output are derived from the first two 4-bit S-Boxes, i.e. from the first byte of the previous state. Using these observations, they crafted four 256-byte lookup tables (1024 bytes in total) that *merge* the S-Box and the permutation layer and as a result, the whole SP network is performed via table lookups.

Performance. The 1024 byte lookup tables eliminate the need for an independent permutation layer, providing us with the fastest available solution. On the downside, we have to perform one lookup for every two bits, resulting in 32 lookups for a 64-bit state (compared to the Squared S-Box that required only 8 lookups for a 64-bit state). Moreover, we need 1024 bytes to store the tables, thus it is not possible to transfer them to the SRAM on our test platform. The theorized 33 % speedup considered with using SRAM instead of flash storage is only possible in an AVR microcontroller with at least 1024 bytes of internal[2] SRAM, for instance ATtiny1634.

Memory Optimizations for Lookup Tables. All the aforementioned solutions with respect to substitution and/or permutation rely heavily on lookup tables. In order to decrease the computational penalty of the table lookups, we performed several code-level optimizations. In Fig. 2 we demonstrate the code required to perform a single table lookup from flash memory.

The lookup operation consists of two `mov` and one `lpm` instruction. Memory is addressed using 16-bit pointers, so the first `mov` loads the high part, while the second `mov` loads the table index that will be accessed.

[1] Benchmarking was performed on ATmega2560.

[2] Additional external SRAM is not an option, since it is at least as slow as flash memory.

Table alignment. We aim to keep the changes required in the high part (`ZH`) to a minimum. Thus, we align the four 256-byte tables required for the merged SP approach in Sect. 3.1 such that they can be accessed by using only the low part (`ZL`) register as an index and keeping `ZH` unchanged. Elaborating, the four lookup tables start from `0x0600, 0x0700, 0x0800, 0x0900` respectively and thus, the 8 high bits of the address part (`0x06, 0x07, 0x08, 0x09`) remain the same while the 8 low bits are sufficient to act as the table index, ranging from 0 to 255 (`0x00` to `0xFF`).

Memory access grouping. Performing two lookup operations in two different tables requires a total of 10 clock cycles (2 `mov` to change the high part of the operation, 2 `mov` to change the index and 2 `lpm` to perform the actual lookup). However, performing two lookup operations on the *same table* requires a total of 8 clock cycles (2 `mov` for the index, 2 `lpm` for the lookup), since the high part of the memory address remains the same. Thus, we we try to perform the maximum amount of grouped table lookups, given the limited number of registers, since within each group, `ZH` remains the same. For instance the following sequence of operations, *lookupTable1(i), lookupTable2(k), lookupTable1(j), lookupTable2(m)* will transform to *lookupTable1(i), lookupTable1(j), lookupTable2(k), lookupTable2(m)* in order to group memory access.

3.2 Key Update Implementation

This section focuses on implementing the key scheduling/update process of the PRESENT cipher efficiently. The key update function of the PRESENT cipher consists of three operations, namely, key rotation, key substitution and key XOR the round counter. We present the optimizations performed in the following subsections.

Key Rotation. The algorithm specifies that the key must be rotated by 61 bits to the left. Given the fact that rotations/shifts are computationally expensive in the AVR architecture, we transform 61 left rotations to 19 right rotations, which can be further reduced to 16 right rotations and 3 right rotations. The 16 right rotations can be easily performed by using the `mov` instructions on register level, i.e. rotate all the bits inside a register by moving the contents to the previous register used in our representation, an approach which is preferable to single rotations via the bit-level instructions. Only the 3 remaining rotations are carried out with the logical instructions for right rotation and shifting (`ror` and `shr`).

Key Substitution. The highest 4 bits of the 80-bit key used by the PRESENT cipher, must be substituted via the S-Box. To avoid 4-bit memory access or redundancy (Sect. 3.1), we construct a special-purpose Squared S-Box that substitutes the 4 high bits of the 8-bit input, while the low 4 bits remain unchanged. The resulting table applies a substitution operation on the upper nibble which takes only a single lookup operation. Should we encounter space constraints, it is possible to replace the Squared S-Box with the original, unpacked one; the key substitution occurs only once per round, so the performance loss incurred by the unpacked S-Box is relatively small.

Key Exclusive-OR Operation. The algorithm specifies that the key bits 15, 16, 17, 18, 19 must be XORed with the round counter. The issue is that -under the current representation- bits 0. . . 7 will be stored in *R0*, bits 8. . . 15 will be stored in *R1* and bits 16. . . 23 will be stored in *R2*. As a result parts of the round counter need to be XORed with different parts of two separate registers, namely the counter needs to be XORed with both *R1* and *R2*. Similarly to Eisenbarth [8], we perform the XOR operation before the key rotation, thus the bits that are operated on are bits 34,35,36,37,38 which span a single register (under the previous representation they are located in *R4*). This restructuring of the algorithm, i.e. performing the XOR operation before the key rotation, does not affect the outcome or security of the algorithm.

Latency *vs.* Throughput. The implementations presented focus on reducing the cost of a *single* encryption. Thus, it can also be viewed as a *low-latency* implementation of PRESENT. Should we loosen up on the latency requirement, we can achieve increased *throughput*. Future work aims to perform multiple encryptions at a time, by using the *bitslicing* technique [3], which largely reduces the cost of permutations.

4 Size-Optimized Implementation

Here we will list some of the size improvements we were able to apply to the PRESENT algorithm. While these modifications make the algorithm operate more slowly, the reduction in size would allow the cipher to be included in microcontrollers with smaller available code area. Our version requires 128 AVR instructions (256 bytes of code) for both the encryption and decryption routines, plus two times 8 bytes for packed tables of S-Box values at memory addresses `0x100` and `0x200`. We believe this should be sufficiently small for the code to be included in an AVR machine with 1K of available Flash storage, while still allowing almost three quarters of the available area to be devoted to application-specific code.

Unfortunately, every opcode in the AVR instruction set is expressed using one or more 16-bits words, so while using only the single-word instructions of the ATtiny there is no further possibility to exchange any instructions for equivalent smaller ones (such as for example *add Rd, 1* being more concisely expressed as *inc Rd* on x86 machines). Furthermore the AVR employs a Harvard architecture, where there is a strict separation between data and code memory; this prevents us from dynamically computing new opcodes in memory to be executed later. Finally we note there are no ‘bulk’ instructions which operate on several registers at once.

However, we do have access to some instructions that are specific to the AVR architecture and are uniquely suited to making parts of the code more condensed by virtue of their expressiveness, such as the *swap* and *cbr* instructions. We also have all kinds of branching instructions at our disposal that branch based on values in the state register (SREG) which we can explicitly or implicitly modify. We can use the stack to make procedure calls or as temporary storage, and finally have the powerful option of adressing the CPU registers indirectly through the Y pointer.

4.1 Serialization and SRAM Use

The greatest size optimization we have implemented is serialization of the algorithm, keeping most of the state in SRAM while we operate on only one byte of state wherever possible. This reduces the instruction count on all parts of the algorithm. It also allows us to make use of fewer dedicated registers.keeping scheduled keys entirely in registers after setup. The only procedure where this approach was not successful is the permutation layer, for which we chose to reserve 4 output registers as we believe it allows us to apply the 4-bit period permutation in software with the most size-efficiency.

By requiring only 4 dedicated registers to keep a partial block state, and 6 to keep the rest of the algorithms' state, we were left with exactly 16 of the 32 general purpose registers available to keep the key register (as we rely on the 6 registers for X,Y,Z to navigate the SRAM, indirectly addressed registers and flash storage respectively.) This means support for 128-bit keys was able to be added to the implementation at no extra cost.

4.2 S-Box Packing

The PRESENT S-Boxes work on 4-bit nibbles, but defining a table of nibbles in the code at first seemed less size-efficient than packing the nibbles into bytes to be unpacked. This would save 8 bytes per S-Box table to start with, but we need 4 to 7 extra instructions depending on whether or not we care about timing attacks to unpack the nibbles which diminishes the size benefit to 2 bytes of code. See Fig. 3 and Table 2.

The S-box construct replaces a single low nibble in the output. The simplest and most size-efficient way to apply it to a byte in code consists of (1) calling it, (2) swapping the resulting nibbles, (3) calling it again for the other nibble and then (4) swapping the nibbles back. We can call this code starting from step 2 to apply the S-box to only the high nibble of a byte, as is required when scheduling new round keys.

4.3 Permutation Layer

The code to apply the bit position permutation to the state in software borrows heavily from the AVR implementation of PRESENT drafted in Louvain by Eisenbarth et al. [8]. Since the permutation follows a 4-bit period in the input, their choice to use 4 bytes of I/O when rotating bits off of registers into corresponding new positions seems quite efficient. In the Louvain implementation one

```
unpack_sBox:
        asr ZL                  ; halve input, take carry
        lpm SBOX_OUTPUT, Z      ; get S-Box output
        brcs odd_unpack         ; branch depending on carry
even_unpack:
        swap SBOX_OUTPUT        ; swap nibbles in S-Box output
        rjmp unpack
odd_unpack:
        nop                     ; guard against timing attacks
        nop
unpack:
        cbr SBOX_OUTPUT, 0xf0 ; clear high nibble in S-Box output
```

Fig. 3. Loading and unpacking bytes into nibbles in a constant cycle count takes us 8 instructions whereas loading unpacked nibbles takes us only 1. The net gain is only 2 bytes of code.

```
block:
        set               ; set T flag
        ; fall through
redo_block:
        ; instructions here happen twice when called from block

        brtc continue     ; break if T flag cleared
        clt               ; clear T flag
        rjmp redo_block   ; redo this block
continue:
        ret
```

Fig. 4. A construct that uses the state register to re-do a block twice with code executing before, after and in between, allowing 2 executions of the block without requiring access to the stack to store return addresses.

bit is rolled off from 4 state registers into one output register and this block is done twice, completing one output byte.

This implementation of the permutation layer requires availability of 4 bytes of temporary storage for half of the permuted state, which we save to the stack before applying the permutation to the other 4 bytes of the state. The construct in Fig. 4 allows the implementer to let a block of code be executed twice, while allowing them to take control of the machine before, after and in between these code blocks. As you can see this takes 4 instructions, whereas a *rcall*, *ret* construction would take us only 3, but this construct doesn't use the stack which means we can keep our intermediate values there rather than requiring 4 more dedicated registers or make more complicated use of the SRAM.

The 4 extra registers that became available through this approach allowed us to implement support for 128-bit keys while keeping the scheduled round keys entirely in CPU registers at no extra size cost. This construct does not affect the cycle count relative to the state or input.

Rather than using specialized code to invert this permutation when decrypting, we use the PRESENT author's design of the bit permutation to undo it by applying it twice more (that is $P(P(P(i))) = i$ for each bit position i).

4.4 Indirect Register Access

The AVR platform allows the CPU registers to be addressed *indirectly*, meaning we can use a pointer (Y) in memory space to interact with them. We use the feature to load all 10 or 16 of the dedicated key registers in a loop, to iterate over them while applying the round key to the state in SRAM, and to rotate the key registers. This last optimization proved to be devastating to performance compared to inlined rotation of the registers, which is why we added an option to configure either approach.

Doing these operations in a loop which iterates over registers results in smaller code, and allows us to rotate an arbitrary number of registers at a fixed instruction cost. Faster (inlined) rotation makes the implementation about 4 times faster and requires 2/8 extra instructions depending on the configured key size.

4.5 Round Key Application and Key Scheduling

The round key is applied to the state in the SRAM by reading, XORing and writing one byte to/from a register at a time. The use of indirect register access allows us to iterate through the key bytes in CPU registers while iterating the state bytes in SRAM.

When scheduling keys, we still apply the exclusive-or to part of the key register in the ideal position (i.e. where the bytes of the key register line up with the round counter register), as explained in Sect. 3.2.

The inverse key scheduling procedure is only needed in the decryption routine, so we were able to inline it into the decryption round.

4.6 Limits Encountered

Although S-Box application and round key application always happen close to each other and have the same SRAM access pattern, the varying order in which they are applied in encryption/decryption rounds, or the omission of the S-Box application in the final step makes it impossible to combine the two steps into one procedure that makes PRESENT smaller.

It is possible to save a few instructions by iterating through the state in SRAM from wherever the X pointer is located rather than rewinding it to the start of the block in every round procedure of the algorithm. Still, the varying order in which they are applied in the encryption/decryption rounds makes it impossible to do so for the general case in smaller code.

The choice to rewind to the start of the block places the SRAM pointer back to the same address as before encryption or decryption, which seems like a good default for real-life use and also allows all round procedures to be callable from external applications, which seemed more desirable than having them rely on ‘hidden’ state which affects their behaviour.

Table 4. Speed (in clock cycles) and size (in bytes) comparisons to existing implementations of PRESENT for the AVR architecture, ordered by size, speed.

	Encryption	Decryption	Size
Papagiannopoulos [20]	8721	-	1794
AVR Crypto-lib [21]	105796	151624	1514
Eisenbarth [8]	10723	11239	936
Verstegen [22]			
Inlined rotation, unpacked S-Boxes (128-bit)	64506	119626	292
Inlined rotation (128-bit)	67854	123346	290
Inlined rotation, unpacked S-Boxes	52622	73952	280
Inlined rotation	55784	77300	278
Unpacked S-Boxes	186883	250032	274
Unpacked S-Boxes (128-bit)	278189	568010	274
Default	190045	253380	272
Default (128-bit)	281537	571730	272

4.7 Using the Code for Specific Applications

While the attained size of our implementation of PRESENT should suffice for use in real-world applications, most of the procedure calls can be inlined when only encryption or decryption is required in the application. If only encryption is required, the inverse S-Box and S-Box unpacking code can be omitted as well.

As mentioned in 4.4, the key register rotation procedure can be configured to either use indirect register addressing, or be inlined at a cost of 4/16 extra bytes depending on configured key size.

5 Conclusion

We've compared our results to the existing AVR implementation by Eisenbarth et al [8] and the GNU AVR-Crypto-Lib (as a standard C reference) [21]. We are pleased to announce we were able to reduce the code size by 70 % and gain 18 % speed increase (Table 4). Having access to a larger SRAM could allow the lookup tables for the speed-optimized version to incur a lower overhead, and reduce an estimated 992 cycles per encryption (12 %). Overall, we managed to push the limits of the PRESENT implementation and establish a wide spectrum of techniques to enable speed and size efficiency.

References

1. Abed, F., Forler, C., List, E., Lucks, S., Wenzel, J.: Biclique cryptanalysis of the PRESENT and LED lightweight ciphers. Technical report, Cryptology ePrint Archive, Report 2012/591 (2012)
2. Anderson, R., Biham, E., Knudsen, L.: Serpent: a proposal for the advanced encryption standard. NIST AES Proposal (1998)

3. Biham, E.: A fast new DES implementation in software. Technion, Technical, Report CS0891 (1997)
4. Bogdanov, A., Knežević, M., Leander, G., Toz, D., Varıcı, K., Verbauwhede, I.: SPONGENT: A lightweight hash function. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 312–325. Springer, Heidelberg (2011)
5. Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: an ultra-lightweight block cipher. In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–466. Springer, Heidelberg (2007)
6. Bogdanov, A., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y.: Hash functions and RFID tags: mind the gap. In: Oswald, E., Rohatgi, P. (eds.) CHES 2008. LNCS, vol. 5154, pp. 283–299. Springer, Heidelberg (2008)
7. Collard, B., Standaert, F.-X.: A statistical saturation attack against the block cipher PRESENT. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 195–210. Springer, Heidelberg (2009)
8. Eisenbarth, T., et al.: Compact implementation and performance evaluation of block ciphers in a tiny devices. In: Mitrokotsa, A., Vaudenay, S. (eds.) AFRICACRYPT 2012. LNCS, vol. 7374, pp. 172–187. Springer, Heidelberg (2012)
9. Gomathisankaran, M., Lee, R.B.: Maya: a novel block encryption function. In: Proceedings of International Workshop on Coding and Cryptography, vol. 33, p. 54 (2009)
10. Gong, Z., Hartel, P., Nikova, S., Zhu, B.: Towards secure and practical MACs for body sensor networks. In: Roy, B., Sendrier, N. (eds.) INDOCRYPT 2009. LNCS, vol. 5922, pp. 182–198. Springer, Heidelberg (2009)
11. Gong, Z., Nikova, S., Law, Y.W.: KLEIN: A new family of lightweight block ciphers. In: Juels, A., Paar, C. (eds.) RFIDSec 2011. LNCS, vol. 7055, pp. 1–18. Springer, Heidelberg (2012)
12. Gong, Z., Zhu, B.: Software implementation of block cipher PRESENT for 8-Bit platforms. http://cis.sjtu.edu.cn/index.php/Software_Implementation_of_Block_Cipher_PRESENT_for_8-Bit_Platforms (2013). Accessed 19 Feb 2013. Archived at http://www.webcitation.org/1370831045860897
13. Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.: The LED block cipher. In: Preneel, B., Takagi, T. (eds.) CHES 2011. LNCS, vol. 6917, pp. 326–341. Springer, Heidelberg (2011)
14. Hirose, S.: Provably secure double-block-length hash functions in a black-box model. In: Park, C., Chee, S. (eds.) ICISC 2004. LNCS, vol. 3506, pp. 330–342. Springer, Heidelberg (2005)
15. Hirose, S.: Some plausible constructions of double-block-length hash functions. In: Robshaw, M.J.B. (ed.) FSE 2006. LNCS, vol. 4047, pp. 210–225. Springer, Heidelberg (2006)
16. Hutter, M., Schwabe, P.: NaCl on 8-bit AVR microcontrollers
17. Information technology - Security techniques - Lightweight cryptography - Part 2: Block ciphers (2011)
18. Leander, G.: Small scale variants of the block cipher PRESENT. IACR ePrint Report, 143 (2010)
19. Nakahara Jr, J., Sepehrdad, P., Zhang, B., Wang, M.: Linear (hull) and algebraic cryptanalysis of the block cipher PRESENT. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 58–75. Springer, Heidelberg (2009)
20. Papagiannopoulos, K.: Speed-optimized implementation of PRESENT in AVR assembly. https://github.com/kostaspap88/PRESENT_speed_implementation/ (2013)

21. The GNU project. AVR-Crypto-Lib. http://avrcryptolib.das-labor.org/ (2013). Accessed 6 April 2013
22. Verstegen, A.: Size-optimized implementation of PRESENT in AVR assembly. https://github.com/aczid/ru_crypto_engineering/ (2013)

Author Index

Aghili, Seyed Farhad 86
Avoine, Gildas 3

Bagheri, Nasour 57, 86
Batina, Lejla 103
Bingöl, Muhammed Ali 3

Carpent, Xavier 3
Ciardulli, Andrea 69

Dalkiliç, Gökhan 19
Das, Amitabh 103
Dubrova, Elena 113

Ege, Barış 103

Gauravaram, Praveen 57, 86

Kardaş, Süleyman 3
Kasper, Timo 34
Kavun, Elif Bilge 103
Kirschbaum, Mario 128
Korak, Thomas 128
Kühn, Alexander 34

Mane, Deepak 147
Mansouri, Shohreh Sharif 113
Mentens, Nele 103

Oswald, David 34
Özcanhan, Mehmet Hilal 19

Paar, Christof 34, 103
Papagiannopoulos, Konstantinos 161

Safkhani, Masoumeh 57, 86
Sanadhya, Somitra Kumar 57, 86
Schaumont, Patrick 147
Sportiello, Luigi 69

Utku, Semih 19

Verbauwhede, Ingrid 103
Verstegen, Aram 161

Wenger, Erich 128

Yalçın, Tolga 103

Zenger, Christian 34