

Advanced Sciences and Technologies for Security Applications

Francesco Flammini *Editor*

Resilience of Cyber-Physical Systems

From Risk Modelling to Threat
Counteraction

 Springer

Advanced Sciences and Technologies for Security Applications

Series editor

Anthony J. Masys, Associate Professor, Director of Global Disaster Management, Humanitarian Assistance and Homeland Security, University of South Florida, Tampa, USA

Advisory Board

Gisela Bichler, California State University, San Bernardino, CA, USA

Thirimachos Bourlai, WVU – Statler College of Engineering and Mineral Resources, Morgantown, WV, USA

Chris Johnson, University of Glasgow, UK

Panagiotis Karampelas, Hellenic Air Force Academy, Attica, Greece

Christian Leuprecht, Royal Military College of Canada, Kingston, ON, Canada

Edward C. Morse, University of California, Berkeley, CA, USA

David Skillicorn, Queen's University, Kingston, ON, Canada

Yoshiki Yamagata, National Institute for Environmental Studies, Tsukuba, Japan

The series *Advanced Sciences and Technologies for Security Applications* comprises interdisciplinary research covering the theory, foundations and domain-specific topics pertaining to security. Publications within the series are peer-reviewed monographs and edited works in the areas of:

- biological and chemical threat recognition and detection (e.g., biosensors, aerosols, forensics)
- crisis and disaster management
- terrorism
- cyber security and secure information systems (e.g., encryption, optical and photonic systems)
- traditional and non-traditional security
- energy, food and resource security
- economic security and securitization (including associated infrastructures)
- transnational crime
- human security and health security
- social, political and psychological aspects of security
- recognition and identification (e.g., optical imaging, biometrics, authentication and verification)
- smart surveillance systems
- applications of theoretical frameworks and methodologies (e.g., grounded theory, complexity, network sciences, modelling and simulation)

Together, the high-quality contributions to this series provide a cross-disciplinary overview of forefront research endeavours aiming to make the world a safer place.

The editors encourage prospective authors to correspond with them in advance of submitting a manuscript. Submission of manuscripts should be made to the Editor-in-Chief or one of the Editors.

More information about this series at <http://www.springer.com/series/5540>

Francesco Flammini

Editor

Resilience of Cyber-Physical Systems

From Risk Modelling to Threat Counteraction

 Springer

Editor

Francesco Flammini
Chair, IEEE SMC Technical Committee
on Homeland Security
Rome, Italy

ISSN 1613-5113 ISSN 2363-9466 (electronic)
Advanced Sciences and Technologies for Security Applications
ISBN 978-3-319-95596-4 ISBN 978-3-319-95597-1 (eBook)
<https://doi.org/10.1007/978-3-319-95597-1>

Library of Congress Control Number: 2018959409

© Springer International Publishing AG, part of Springer Nature 2019

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

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

To My Beloved Daughter, Marina

Foreword

Cyber-physical systems (CPS) study the close interaction and the deep integration between cyber information systems and dynamic physical systems. Those physical systems could be energy systems, automotive systems, human activities, surrounding environment, etc. The key is to identify the deep underlying links between the information system and the target physical system. In particular, CPS security studies the security issues in such links.

The formulations of CPS security problems emphasize the impacts to physical systems resulting from the attacks to information systems. For example, a smart grid is usually associated with thousands of information nodes. Only small portions of those information nodes are “critical,” which means that hacking them could lead to large impact to the power systems. The classical cybersecurity usually deals only with the information system. It does not distinguish critical information nodes from noncritical information nodes in terms of the physical impacts, which is however important in many real-world applications. In contrast, CPS security research jointly models the dynamics of the information system and the physical system and evaluates the cyberattack impacts. It heavily interleaves the security studies in both systems, which is the key to the interdisciplinary “cyber-physical” security research.

In terms of developing attack detection techniques, typically an incomplete physical model and a set of data are given. The target is to make the incomplete physical model “more complete” and more useful. In other words, one needs to develop the confluence of physical model-based approach and data-driven approach, which characterizes the “cyber-physical” solution. Finding the underlying links between the information system and the physical system to build such a cyber-physical solution is difficult. In addition, the induced computational complexity, scalability, efficient updating, threat modeling, and behavior modeling issues impose significant challenges as well.

This book addresses the issue of resilience in the context of cyber-physical systems. Resilience is the ability to provide and maintain an acceptable level of service in the face of faults and challenges to normal operation. Resilience is a

composed dependability attribute that is much stronger and more dynamic than reliability, safety, or security, although it includes them, together with concepts like threat tolerance and self-healing. The chapters included in this book deal with methodological aspects related to resilience modeling and evaluation, including risk assessment, as well as technologies and tools for implementing effective resilience in real applications from diverse and current domains. As such, I believe this book is an excellent source for students, researchers, and practitioners needing to investigate issues related to resilient cyber-physical systems in order to analyze vulnerabilities and mitigate consequences through not only prevention but also active response and dynamic reconfiguration.

Chair of Cyber-Physical Systems
Linnaeus University
Växjö, Sweden

Professor Shiyan Hu, Ph.D.

Preface

In the last years, we have witnessed an acceleration in the adoption of computer systems in all societal sectors. Experts have started figuring out many new definitions and paradigms to catch the crucial characteristics of modern computing, including pervasive computing, ubiquitous computing, etc. Now, we are entering a computing era in which, together with aspects like size, distribution, and heterogeneity, other complexity factors have become to emerge, mainly related to autonomy and symbiosis. In other words, novel challenges are arising when diverse connected IoT (Internet of Things) smart devices start to cooperate autonomously with each other, and with humans, by leveraging on artificial intelligence and machine-learning techniques. Those challenges are especially critical in domains, like Industry 4.0 and intelligent transportation systems, where the cyber and physical worlds merge to create almost indistinguishable electromechanic entities (i.e., cyber-physical systems) needing holistic analysis and control. Among all the challenges, the ones addressing reliability, safety, and security are the most critical. On that regard, the scientific community decided to extend and update the traditional taxonomy of embedded real-time systems and dependable computing in order to cope with the aforementioned paradigms of modern computing. One of the attributes that best captured the evolving dependability requirements of cyber-physical systems is the one known as “resilience.” As many other bio-inspired paradigms, the word resilience brilliantly catches the capacity to adapt/react, reconfigure, and self-heal required to symbiotic autonomous cyber-physical systems.

Together with the societal, ontological, and technological implications of such a rapid evolution, risk analysis remains a central theme that is still essential to drive the design, development, and management of cyber-physical systems. Nowadays, one of the main challenges with risk analysis is to develop scalable model-based and model-driven approaches providing quantitative results that are suitable to a large class of different systems featuring a growing complexity. Furthermore, with Internet-connected devices, it is essential to assess vulnerabilities in a continuous cycle, in order to cope with the most recent cyber-security threats.

It is clear that in such a scenario, control systems and information infrastructures would greatly benefit from techniques automating the whole risk management process, or at least part of it. To tackle this challenge, I do believe future efforts in research and innovation must go in the direction of investigating proactive resilience, merging threat tolerance and artificial intelligence in all their possible declinations, with the potential of creating the next generation of resilient cyber-physical systems.

This book explores some of the aforementioned issues by addressing resilience in the context of cyber-physical systems. It consists of three parts, each one including three chapters.

The first part of the book addresses “Challenges and Frameworks” to achieve resilience in cyber-physical systems.

The first chapter of Part I by Dániel Tokody, József Papp, László Barna Iantovics, and Francesco Flammini, entitled “Complex, Resilient and Smart Systems,” introduces the reader to the context and theoretical concepts of smart cyber-physical systems as well as to novel paradigms of machine intelligence enabling future-generation resilience.

In the second chapter of Part I by Marco Rocchetto, Alberto Ferrari, and Valerio Senni, entitled “Challenges and Opportunities for Model-Based Security Risk Assessment of Cyber-Physical Systems,” the authors address in an extensive survey the issue of model-based security risk assessment in modern cyber-physical systems from most current and critical industry domains, highlighting state-of-the-art methods and reference standards.

The third chapter of Part I by Hassan Mokalled, Concetta Pragliola, Daniele Debortol, Ermete Meda, and Rodolfo Zunino, entitled “A Comprehensive Framework for the Security Risk Management of Cyber-Physical Systems,” continues along the lines set by the topics of the previous chapter. It proposes a holistic framework for systematic risk management applied to IT systems validated in a relevant industrial application of railway supervision, monitoring, and control systems.

The second part of the book presents “Evaluation Methodologies and Tools” for resilience modeling and assessment of cyber-physical systems.

The first chapter of Part II (4th book chapter) by Janusz Górski and Andrzej Wardziński, entitled “Supporting Cybersecurity Compliance Assessment of Industrial Automation and Control System Components,” introduces a tool for compliance assessment against cybersecurity standards applicable to industrial automation and control systems, through a clear and very effective running example.

In the second chapter of Part II (5th book chapter) by Oleksandr Netkachov, Peter Popov, and Kizito Salako, entitled “Quantitative Evaluation of the Efficacy of Defence-in-Depth in Critical Infrastructures”, the authors report on a model-based approach in assessing cyber-risks by deploying a defense-in-depth approach implementing the so-called pro-active recovery of protection devices in order to ensure resilience against threats.

In the third chapter of Part II (6th book chapter) by Frederik Gossen, Tiziana Margaria, Johannes Neubauer, and Bernhard Steffen, entitled “A Model-Driven and

Generative Approach to Holistic Security with C-IME,” the authors present a cyber resilience tool allowing for automatic generation of security features through proper model-driven engineering and domain-specific languages.

The third part of the book describes relevant “Industrial Applications” of electrical distribution, water supply, and nuclear plants, focusing on specific approaches and countermeasures.

In the first chapter of Part III (7th book chapter) by Julian L. Rrushi, entitled “Multi-range Decoy I/O Defense of Electrical Substations Against Industrial Control System Malware,” the author addresses a specific application of electrical substations needing protection against industrial control systems malware. He provides the specific case study description and related results in extensive details.

In the second chapter of Part III (8th book chapter) by Fabio Tarani, Chiara Arrighi, Laura Carnevali, Fabio Castelli, and Enrico Vicario, entitled “Flood Resilience of a Water Distribution System,” the authors tackle the specific challenge of ensuring the resilience of a computer-controlled water supply system against flooding risks.

In the third chapter of Part III (9th and last book chapter) by Wei Wang, Francesco Di Maio, and Enrico Zio, entitled “A Non-parametric Cumulative Sum Approach for Online Diagnostics of Cyber Attacks to Nuclear Power Plants,” the authors face the issue of managing cyberattacks to nuclear plants. In such a context, they focus on diagnostics to drive protection and mitigation actions and mathematical/probabilistic modeling of system resilience, providing several interesting and useful results.

It is important to underline that each chapter in this book has been peer-reviewed prior to final acceptance. Authors have been requested to take into account any concerns and to implement all the suggestions provided by the reviewers.

I would like to take this opportunity to express my sincere gratitude to the authors, to the members of the Editorial Advisory Board, to the additional reviewers, and to anyone who has helped me in book development and quality improvement, for their invaluable contributions and support.

Special thanks go to my beloved family, Liana and Marina, for being so tolerant (and resilient!) against my cyber-nerd attitude and variable mood.

Rome, Italy

Francesco Flammini

Contents

Part I Challenges and Frameworks	
Complex, Resilient and Smart Systems	3
Dániel Tokody, József Papp, László Barna Iantovics, and Francesco Flammini	
Challenges and Opportunities for Model-Based Security Risk Assessment of Cyber-Physical Systems	25
Marco Rocchetto, Alberto Ferrari, and Valerio Senni	
A Comprehensive Framework for the Security Risk Management of Cyber-Physical Systems	49
Hassan Mokalled, Concetta Pragliola, Daniele Debertol, Ermete Meda, and Rodolfo Zunino	
Part II Evaluation Methodologies and Tools	
Supporting Cybersecurity Compliance Assessment of Industrial Automation and Control System Components	71
Janusz Górski and Andrzej Wardziński	
Quantitative Evaluation of the Efficacy of Defence-in-Depth in Critical Infrastructures	89
Oleksandr Netkachov, Peter Popov, and Kizito Salako	
A Model-Driven and Generative Approach to Holistic Security	123
Frederik Gossen, Tiziana Margaria, Johannes Neubauer, and Bernhard Steffen	
Part III Industrial Applications	
Multi-range Decoy I/O Defense of Electrical Substations Against Industrial Control System Malware	151
Julian L. Rrushi	

Flood Resilience of a Water Distribution System 177
Fabio Tarani, Chiara Arrighi, Laura Carnevali, Fabio Castelli,
and Enrico Vicario

**A Non-parametric Cumulative Sum Approach for Online
Diagnostics of Cyber Attacks to Nuclear Power Plants** 195
Wei Wang, Francesco Di Maio, and Enrico Zio

Index 229

Editorial Advisory Board

Cristina Alcaraz	University of Malaga, Spain
Kamel Barkaoui	Cedric – CNAM, France
Cinzia Bernardeschi	University of Pisa, Italy
Simona Bernardi	University of Zaragoza, Spain
Sandro Bologna	AIIC – Italian Association of Experts in Critical Infrastructures, Italy
Andrea Bondavalli	University of Florence, Italy
Michael Butler	University of Southampton, UK
António Casimiro	University of Lisbon, Portugal
Mirko D’Angelo	Linnaeus University, Sweden
Felicita Di Giandomenico	ISTI-CNR, Italy
Alessandro Fantechi	University of Florence, Italy
Luca Faramondi	University of Rome Campus Bio-Medico, Italy
Francesco Flammini	Linnaeus University, Sweden
Lenzini Gabriele	SnT - University of Luxembourg, Luxembourg
Barbara Gallina	Mälardalen University, Sweden
Rick Harper	IBM, USA
Mansur Hasib	University of Maryland, USA
Leonardo Impagliazzo	Ansaldo STS, Italy
Mohamed Kaaniche	LAAS-CNRS, France
Aron Laszka	University of Houston, USA
Qiudan Li	Chinese Academy of Sciences, China
Nicola Mazzocca	University of Naples Federico II, Italy
Gyula Mester	Óbuda University, Hungary
Takashi Nanya	University of Tokyo, Japan
András Pataricza	Budapest University of Technology and Economics, Hungary
Peter Popov	City University of London, UK

Paolo Prinetto	Polytechnic University of Turin, Italy
Akshay Rajhans	MathWorks, USA
Luigi Romano	University of Naples Parthenope, Italy
Carlo Rusconi	SOGIN – Nuclear Plants Management Society, Italy
Francesca Saglietti	University of Erlangen-Nuremberg, Germany
Karsten Speckmann	NCISS – NATO Communications and Information Systems School, Italy
Neeraj Suri	Technical University of Darmstadt, Germany
Daniel Tokodi	Óbuda University, Hungary
Stefano Tonetta	FBK – Bruno Kessler Foundation, Italy
Enrico Vicario	University of Florence, Italy
Marco Vieira	University of Coimbra, Portugal
Valeria Vittorini	University of Naples Federico II, Italy
Edgar Weippl	SBA Research, Austria
Stefano Zanero	Polytechnic University of Milan, Italy
Quanyan Zhu	New York University, USA
Armin Zimmermann	Technical University of Ilmenau, Germany

Additional Reviewers

K

Khakpour, Narges

M

Mester, Gyula

P

Pogliani, Marcello

S

Senni, Valerio

Part I
Challenges and Frameworks

Complex, Resilient and Smart Systems



Dániel Tokody, József Papp, László Barna Iantovics, and Francesco Flammini

Abstract “Cyber-Physical Systems or “smart” systems are co-engineered interacting networks of physical and computational components. These systems will provide the foundation of our critical infrastructure, form the basis of emerging and future smart services, and improve our quality of life in many areas.” (National Institute of Standards and Technology: Cyber-physical systems. [Online]. Available: <https://www.nist.gov/el/cyber-physical-systems>. Accessed 31 Dec 2017, 2017). The concept of Smartness has been increasingly used as a marketing catchphrase. This study seeks to explain that smartness can be a serious indicator which can help to describe the machine intelligence level of different devices, systems or networks weighted by, among others, the usability index. The present study aims to summarize the implementation of complex, resilient and smart system on the level of devices, systems and complex system networks. The research should consider a smart device as a single agent, the system as a multi-agent system, and the network of complex systems has been envisaged as an ad hoc multi-agent system (Farid AM: Designing multi-agent systems for resilient engineering systems. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), vol 9266, pp 3–8, 2015) organised in a network. The physical incarnations of this latter could be, for example, the subsystems of a smart city. In order to determine the smartness of a certain system, the Machine Intelligence Quotient (MIQ) (Iantovics LB, Gligor A, Georgieva V: Detecting outlier intelligence in the behavior of intelligent coalitions

D. Tokody (✉) · J. Papp

Doctoral School on Safety and Security Sciences, Óbuda University, Budapest, Hungary
e-mail: tokody.daniel@dosz.hu; papp.jozsef@kvk.uni-obuda.hu

L. B. Iantovics

Petru Maior University, Tirgu Mures, Romania
e-mail: Iantovics3ibarna@science.upm.ro

F. Flammini

Linnaeus University, Växjö, Sweden
e-mail: francesco.flammini@lnu.se

of agents. In: 2017 IEEE congress on evolutionary computation (CEC), pp 241–248, 2017; Park H-J, Kim BK, Lim KY: Measuring the machine intelligence quotient (MIQ) of human-machine cooperative systems. *IEEE Trans Syst Man Cybern – Part A Syst Humans* 31(2):89–96, 2001; Park HJ, Kim BK, Lim GY: Measuring machine intelligence for human-machine cooperative systems using intelligence task graph. In: *Proceedings 1999 IEEE/RSJ international conference on intelligent robots and systems. Human and environment friendly robots with high intelligence and emotional quotients (Cat. No.99CH36289)*, vol 2, pp 689–694, 1999; Ozkul T: Cost-benefit analyses of man-machine cooperative systems by assesment of machine intelligence quotient (MIQ) gain. In: 2009 6th international symposium on mechatronics and its applications, pp 1–6, 2009), Usability Index (UI) (Li C, Ji Z, Pang Z, Chu S, Jin Y, Tong J, Xu H, Chen Y: On usability evaluation of human – machine interactive Interface based on eye movement. In: Long S, Dhillon BS (eds) *Man-machine-environment system engineering: proceedings of the 16th international conference on MMESE*. Springer, Singapore, pp 347–354, 2016; Szabó G: Usability of machinery. In: Arezes P (ed) *Advances in safety management and human factors: proceedings of the AHFE 2017 international conference on safety management and human factors*, July 17–21, 2017, The Westin Bonaventure Hotel, Los Angeles, California, USA. Springer International Publishing, Cham, pp 161–168, 2018; Aykin N (ed): *Usability and internationalization of information technology*. Lawrence Erlbaum Associates, Inc., Publishers, Mahwah, 2005) and Usability Index of Machine (UIoM), Environmental Performance Index (Hsu A et al: *Global metrics for the environment*. In: *The environmental performance index ranks countries’ performance on high-priority environmental issues*. Yale University, New Haven, 2016) of Machine (EPIoM) indexes will be considered. The quality of human life is directly influenced by the intelligence and smart design of machines (Farid AM: *Designing multi-agent systems for resilient engineering systems*. In: *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)*, vol 9266, pp 3–8, 2015; Liouane Z, Lemlouma T, Roose P, Weis F, Liouane Z, Lemlouma T, Roose P, Weis F, Neu HMAG: A genetic neural network approach for unusual behavior prediction in smart home. In: *Madureira AM, Abraham A, Gamboa D, Novais P (eds) Advances in intelligent systems and computing*, vol 2016. Springer International Publishing AG, Porto, pp 738–748, 2017). Smartness of systems have an indispensable role to play in enabling the overall resilience of the combined cyber-physical system.

1 Introduction: From Automated Machine to Autonomous Smart Machine

There is a global demand for maintaining or improving the quality of life of citizens and increased efforts are being made at all levels in order to fulfil such a demand. Some regions are becoming overpopulated while others suffer from depopulation, and, despite contrary efforts, there is a growing inequality in the distribution of

resources. Technological development and automation have had a complex effect on the whole mankind. Automation has a significant importance for the society, as it can help to reduce human factors in different processes to such an extent which ensures greater resilience, usability and cost-efficiency in various sectors of both industrial production and everyday life. Automation is not simply a technical question; it is also an economic and social issue which concerns the whole society. A fully-automated society saves humans a considerable amount of time which can be used for scientific and cultural purposes or other useful activities. The primary aim of this chapter is to discuss the present and future role of automation in human life. It will highlight how to reach early automation systems to smart autonomous cyber-physical systems.

What are the technical and economic advantages of automation? Improved living standards, workforce efficiency, energy saving, cost reduction, better working conditions, health protection, quality improvement, reduced amount of rejects, increased operational safety and reliability, etc.

How long has automation been present and what does it mean exactly? The roots of automation can be found in the word “automata”. This word originates from the Greek language and means self-operating or self-moving. No one knows exactly who made the first “automata” and when it was made. It is probable that the first structure of this type was built in ancient times as a result of much-feared sorcery and parallel scientific developments. In this context, medieval legendary mentions the name of Albertus Magnus (later: Saint Albert the Great). As a major scientist of his age, he was referred to as “Magnus in magia, major in philosophia, maxmus in theologia” or “Doctor Universalis”. Legend says that he was working for many decades on the construction of a mechanic servant. The fate of the construction, however, was doomed when one of his apprentices, who was received by the “servant” and heard it say to wait for the master, claimed that the machine must be the devil’s invention. Other legends have a different ending. According to these records Magnus’ apprentice, Thomas Aquino (later Saint Thomas Aquinas) destroyed the work of his master. It is also interesting to mention that Albertus Magnus is, among others, the patron saint of scientists [12].

The path leading to full automation or smart autonomous systems has been lined with many milestones in the course of time. Such milestones were the first steam engines and centrifugal governors, then the emergence of controllable electric machines and the appearance of computers, whose future role in automation is difficult to predict as yet. In other words, these human-made devices have been aimed to tame first the power of steam, then electricity and finally information.

The use of machines and the application of automation eliminate inappropriate or inadequate human factors in the processes where their use has been justified. Due to the fact that machines lack self-awareness, they can perform the tasks allocated to them at their best knowledge, the fastest and most efficient way. They will not be affected by ignorance, tiredness or any other human or physiological needs. As long as the conditions necessary for their functioning are ensured, they will accomplish their tasks. Full automation can be achieved first by partial and then by complex mechanisation.

From the viewpoint of automation, different levels can be distinguished. In case of partial automation, the system cannot function without human presence in the measuring, supervising and controlling roles. Without human contribution they are unable to operate. In case of complex automation, these roles are completely overtaken from humans by the automated systems. With fully self-operating systems only the tasks of supervision, development, checking, repair and maintenance will be performed by humans [12].

In the 1970s, the importance of automation was recognised from the point of industrial use in the increase of productivity. Today its significance has far exceeded its industrial role, as automation has been introduced in all fields of life. For example, by the development of nanorobots [13] and their use in the medical treatment of animals and humans, automation provides great opportunities for the society. Not to mention another exciting area of automation represented by robots (e.g. UAVs [14] or industrial robots [15]) or human-like robots, in other words, androids.

From the point of humanity, the significance of automation can be compared to the significance of the industrial revolution, or rather, mechanisation and automation can be regarded as the stages of development that finally determined the industrial revolution [12].

Wolfgang Wahlster, Director of the German Research Centre for Artificial Intelligence, already talked about Industry 4.0 in a presentation in 2013. According to his view we live in the age of the 4th industrial revolution that in terms of industrial automation, means the implementation of cyber-physical systems [16].

Cyber-physical systems will play an increasingly important role in automation, but not only in industrial automation. This new industrial revolution will affect many other fields of science.

Our life has been automated to such level which, for example, allows for autonomously operating vehicles to get from point A to point B safely (e.g. sensor-based intelligent mobile robot navigation [17]). Of course, these systems also require specific conditions and have their own limitations, but they are able to operate in an increasingly automatic way to perform a growing number of tasks [18].

But what is the exactly a smart machine? “A smart machine is a device embedded with machine-to-machine (M2M) and/or cognitive computing technologies such as artificial intelligence (AI), machine learning or deep learning, all of which it uses to reason, problem-solve, make decisions and even, ultimately, take action” [19].

2 Usability, Environmental Performance, Smartness and Resilience

What is Smart Machine design? At the design stage of the system, it is necessary to quantify the system parameters and check the main system features as early as possible. In general, it can be said that in order to develop a practicable, flexible

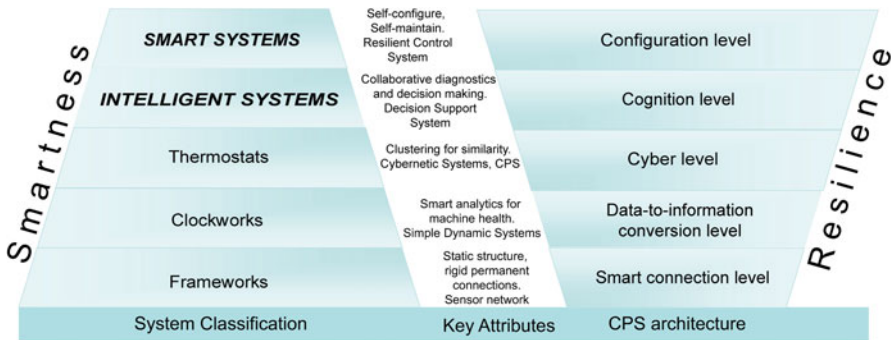


Fig. 1 5C architecture for design of smart cyber-physical systems [21, 22]

and resilient system, the aspects of scalability, modularity and extensions must be taken into consideration. There is a growing tendency of considering such subjective parameters as elegance or the comfort level ensured by the system. Ideally at the planning phase the possibilities of creating a future-proof design which allows the system to be used for future purposes should also be considered. In the course of such planning it is important to bear in mind the rapidly changing technological developments as well as the compliance with various international standards. With regard to economy, energy efficiency and other requirements for such systems, it is necessary to identify the bottlenecks hidden in the system. A system will always have some bottlenecks, either because of economic conditions or the limited availability of resources. If such bottlenecks are planned, however, the optimality of the system can be ensured and a harmonious system can be developed [20]. (see Fig. 1. 5C architecture for implementation and design [21]).

What is resilience? According to the Cambridge English Dictionary, the resilience is the quality of being able to return quickly to a previous good condition after problems. How can smartness support future-proof resilience? Smartness is the road to future-proof resilience since it enables novel paradigms like proactive dependability and self-healing. Those paradigms are completely different from the current implementations of safety-critical and dependable systems where threats, vulnerabilities and consequences are supposed to be known in advance during risk assessment or somehow updated and uploaded later (e.g. threat signatures/patterns, vulnerability information, troubleshooting instructions and repair workflow, etc.). Resilience in future smart-systems will increasingly leverage on embedded intelligence in order to anticipate and detect unknown threats and automatically compute the most appropriate and safe solutions by using approaches based on machine learning, heuristics, fuzzy logic, bayesian inference and artificial neural networks driving real-time co-simulation and online model-checking.

Smart machines represent a new field which is still to be thoroughly researched. Recent research shows that the creation of smart machines is bringing to a higher level of automation in many fields and that trend will continue in the future.

We mentioned that automation can be a tool to improve resilience. In terms of functionalities, this automation level is close to robotisation, although there are some differences. To make things more complex, the usage of smart systems in critical infrastructures is actually generating the new paradigm of “Smart Critical Infrastructures”, whose resilience evaluation becomes even more challenging. “The resilience of an infrastructure is the ability to anticipate possible adverse scenarios/events (including the new/emerging ones) representing threats and leading to possible disruptions in operations/functionality of the infrastructure, prepare for them, withstand/absorb their impacts, recover from disruptions caused by them and adopt to the changing conditions.” [23] There are five main resilience engineering phases in protecting smart critical infrastructures: understand risk, anticipate or prepare, absorb/withstand, respond/recover, adapt/learn [24]. Adaptation/learning must leverage on machine intelligence.

Machine intelligence was first mentioned probably in the 1940s. With the origin of cybernetics, the possibilities to create thinking machines and robots were reconsidered. The creation of smart machines is based on the development of machine intelligence. Among other trends, connectionism has also had a great influence on the development of smart machines [25].

How could it be possible to create machines which would solve the problems of humanity? The planning and development of smart machines are often limited by human abilities. As it is the case, for example, with the way of thinking software developers acquire at school. In fact, today software development is based on the well-established methods learnt at school according to widely accepted conventional paradigms. Different paradigms are not combined until software developers form their own style. One of the primary aims of software development is to make these programs error-free. It is obvious that errors cannot be completely eliminated; still software developers will always aim to find new improved solutions for the given tasks. Would all this guarantee problem-free operation? Even if it was possible to develop a perfect program, one question would still remain to answer. How could such a rigid system fit into a world that is continuously changing? How human or natural processes adapt to new or unknown situations? Could this be implemented in the system of machines? If humans meet new problems while performing their tasks, they will try to find new solutions to solve them. Today’s machines are rarely capable of doing something similar. It is enough to consider the example of autonomous self-driving vehicles, which represent one of the most advanced and discussed modern technologies. If a self-driving vehicle equipped with state-of-the-art artificial vision approaches a lorry, the back of which is covered with a picture of a group of cyclists, the vehicle will interpret this image as a different traffic situation. For humans, it is not a problem to recognise the situation, as they are able to perceive more details and make various decisions based on formerly learned patterns, as they will know which pattern would work in this situation. Similar considerations make today’s most advanced smart-systems still seem rather “stupid” when compared with basic human intelligence, adaptation and problem-solving capabilities.

2.1 Usability Index and the Usability Index of Machine – UIoM

What is Usability?

The word usability means that the people that use a product can do so quickly and easily to accomplish their own tasks. This definition comes up from four points, which are essential for understanding what usability really are.

1. Focusing on the user
2. People use products to be productive
3. Users are busy people trying to accomplish tasks
4. The user decides when a product is easy to use

This means that to develop a usable product it's important that the developer know, or understand, how the potential user works. No one can fully replace the potential user, hence the first point. [9]

In order to determine the Usability Index, it is necessary to consider aspects of multiple research fields at the same time. One of these fields is ergonomics, which aims to understand human capacities and abilities. The other field is Human-Computer Interaction, which deals with the planning, implementation and evaluation of computer systems used by humans. The third aspect is User Experience, which refers to the expectations and the real experience humans have while using a product, a device or a machine from the date of purchase to their disposal. This will include the shopping experience, the helpdesk support provided during the use of the product, the convenience, luxury and the feeling that is associated with the product. The fourth field is User-centred design, a planning process which integrates the user's expectations and demands. In many cases this also includes the involvement of the user in the planning process. The final aspect to be considered is Usability, which measures the effectiveness, efficiency and satisfaction the user experiences while performing a specific task with the help of the product in a given environment [26, 27].

The criteria referring to the general usability design of machines (see Fig. 2 Usability Engineering process) can be determined on the basis of the information provided by a much researched field. This is the field of the usability design of medical devices. The process of the usability design of machines consist of ten steps [26, 27].

Step 1 (User research): the Application specification contains the concept of usage, the users and operators, the connections of the device to other devices, the conditions of its use and the basic principles of operation. Such a specification must be based on a user survey and market research carried out in advance. The developers' team must have a coherent vision of the device to be developed and its long-term purposes in order to understand the basic requirements for the developed product [26, 27].

Step 2 (Conceptual design): the Frequently used functions are specified in the course of the theoretical design, in order to define the most frequently used and most important functions of the device [26, 27].

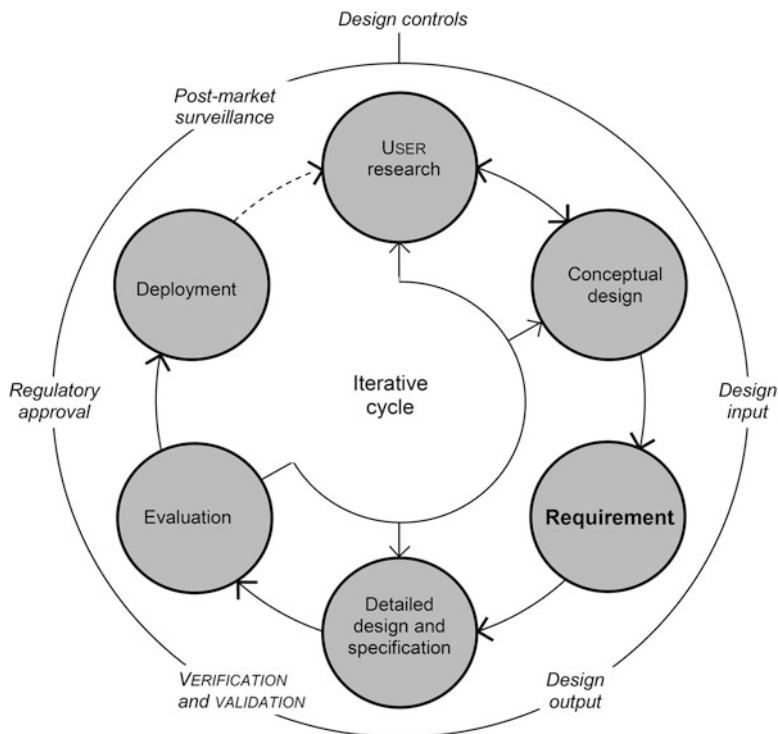


Fig. 2 Usability engineering process (© IEC 62366:2007, Fig. D.1) [27]

Step 3 (Conceptual design): the conceptual design deals with the Identification of hazards and hazardous situations related to usability. User activities and the use of the device – with the exception of fully autonomous devices – can generate errors. The prior identification and treatment of errors is done as part of the risk-management procedure which also affects this phase of the usability process. Therefore, it is necessary to define the criteria for the safe use of the device and to identify the hazards which could be detected in advance [26, 27].

Step 4 (Requirement): furthermore, the Primary operating functions and requirements must be determined. Among the frequently used functions, safety critical functions are those functions which are the most critical from the point of hazardousness.

Step 5 (Requirement): the second phase of the elaboration of requirements is Usability specification, which summarises the information collected about the device in the previous phases, with special attention made to primary operational functions. This is the basic document of the certification and validation of usability [26, 27].

Step 6 (Requirement): the third phase of the elaboration of requirements is the development of the Usability validation plan. The Usability validation plan must

be prepared prior to the validation procedure, as this document specifies the validation methods, validation criteria and the examinations carried out with the involvement of representative users [26, 27].

Step 7 (Detailed design and specification): in this phase a detailed design must be provided which includes the User interface design and implementation. It should also include software development, the making of prototypes and the simultaneous evaluation of usability [26, 27].

Step 8 (Usability verification): The evaluation of the created device starts with the Usability verification. This means the comparison of the device with its specifications and purposes. It ensures the product's compliance with its requirements [26, 27].

Step 9 (Usability validation): Usability validation is the second phase of the evaluation, in which the compliance with user requirements is analysed according to the validation plan. The methods used in the usability evaluation process include analytic, empiric, formative or summative methods, which help to reveal any usability issues [26, 27].

Step 10 (Monitoring): Usability issues can prevent the device from performing its task or fulfilling its purposes. It can cause uncertainty in the users, who might make a mistake as a consequence (e.g. they might fail to notice something, make incorrect assumptions, perform inappropriate actions or misinterpret some information). The surveillance and monitoring of the device can be done after it has been introduced into the market. Any feedback from the users may help to correct the errors [26, 27].

The applied main components are listed in Table 1 with reference to the Usability Index of Machine (UIoM). From the discussion in this section, it should be clear that usability is strictly related to the interference of human-factors with attributes like performability and resilience, since e.g. bad design of Human-Machine interfaces can facilitate human faults. Human faults can generate errors and then failures when those errors activate. Human mistakes and response delays can also significantly impact on the threat/crisis management workflow in business continuity, disaster recovery and reaction to emergencies. Generally speaking, since human actions are error-prone, wherever total automation cannot be achieved yet and human intervention is still required, those aspects are of paramount importance in cyber-physical systems design.

2.2 Environmental Performance Index of Machines – EPIoM

Environmental Performance Index of Machines was developed as a result of the co-operation of Yale University, Columbia University, the World Economic Forum and the European Commission and it was first published in 2002 [10]. This study refers to the EPI version which is related to machines. In case of machines it is especially important to reduce the use of non-renewable resources and to

Table 1 The origin of smartness quotient, the revised MIQ [own]

		Smartness quotient				EPIOM indicators	
		Machine intelligence quotient				UIOM indicators	
Dimensions/ Key attributes	Components	1. Dim. Autonomy	2. Dim. Human-machine interaction	3. Dim. Controllability for complicated dynamics	4. Dim. Bio-inspired behavior	5. Dim. Usability index of machine	6. Dim. Environmental performance index of machine
	Self-calibration		Human-like understanding communication	Non-conventional	Neuro-science, neural networks	Be easy-to-use	Life cycle time
	Self-diagnostics		Emergence of artificial emotion	Model-based	Biologically motivated-behavior	Product quality	Environmental risk exposure
	Self-tuning		Ontology based	Adaptation	Cognitive-based	Innovativity	Household air quality
	Fault tolerance		Auto back-up	Motion planning	Ant colony optimization	User experience	Air pollution – average exposure to PM2.5
	Omniscience		Character “personality and emotional state”	Non-linearity	Evolutionary algorithm	Applicability	Airpollution-PM2.5 exceedance
	Self-contained		Symbiosis	Observability	Self-organizing	Ergonomic design	Air pollution – average exposure to NO ₂
	Reactive		Ambient	Cross-platform	Decentralized	Effectiveness	Unsafe sanitation
	Pro-active		Physical contact	Geometrical	Semantic memory	Efficiency	Drinking water quality
	Rationally		Psychological contact	Structural controllability	Stochastic processes	Satisfaction	Wastewater treatment
	Learning		Spiritual contact	Complexity	Random processes	Ergonomics	Nitrogen use efficiency

(continued)

Table 1 (continued)

Smartness quotient		Machine intelligence quotient		EPIOM indicators		UIOM indicators	
Dimensions/ Key attributes	1. Dim. Autonomy	2. Dim. Human-machine interaction	3. Dim. Controllability for complicated dynamics	4. Dim. Bio-inspired behavior	5. Dim. Usability index of machine	6. Dim. Environmental performance index of machine	
	Goal-oriented	Dialogue systems	Feedback	Problem solving	Understandability	Nitrogen balance	
	Collaborative	Control devices	Stocks, flows	Self-organization technique	Learnability	Change in forest cover	
	Flexible	Visual displays	Saturation	Adaptive heuristic search algorithm	Operability	Fish stocks	
	Self-starting	Adaptive interfaces.	Causal loop diagram	Bio-inspired tactics	Attractiveness	Terrestrial protected areas (national biome weights)	
	Temporal continuity	Auditory displays	Genericity	Optimization solvers	Maintainability	Terrestrial protected areas (global biome weights)	
	Communicative	Knowledge-based support	Energy-related	Genetic algorithm	Functionality	Marine protected areas	
	Adaptive	Procedural support	Modularity, submodularity	Foraging behaviour	Reliability	Species protection (national)	
	Mobile	Fault management support	Self-dynamics	Odometry	Efficiency	Species protection (global)	
	Fault prevention	User-oriented and application-oriented functionalities	Solution based	Swarm intelligence	Portability	Trend in carbon intensity	
Fault removal	Safety	Path planning	Tele-operation	Quality	Trend in CO ₂ emissions per KWH		
Fault forecasting	Human factors	Stability	Collectivity (collective behavior)	Safety	access to electricity		

benefit from renewable energy. They must be designed in a way that considers the optimisation of the use of raw materials and the recycling of future (electronic) waste, in order to ensure the environmentally conscious use of the device in its whole lifecycle [28, 29]. The applied main components are listed in Table 1 with reference to EPIoM.

3 The Implementation of Complex, Resilient and Smart Systems

3.1 Smart Device – Smart Agent

What is a smart agent used for? And what makes it smart? A smart agent (see Fig. 3a) has artificial intelligence, and it stores the information necessary for its operation in ontologies by means of knowledge representation and semantic models. It can also gain knowledge by learning. Supported by this knowledge, it can recognize and even change the surrounding environment. It is capable of independent operation, but it can also work as part of a system with cooperation and self-organization capabilities [3, 30–32].

Smart systems are based on some forms of machine intelligence; therefore, in order to create smart machines, it is necessary to know how develop machine intelligence. Wechsler defined the concept of intelligence as the following: “Intelligence is the aggregate or global capacity of the individual to act purposefully, to think rationally and to deal effectively with his environment” [42]. This definition has been given with regards to human intelligence. However, it is also relevant for machines, as the basis of machine intelligence also includes environment detection, decision making and intervention control. At a higher level, intelligent machines are capable of recognising objects and events, representing knowledge or determining

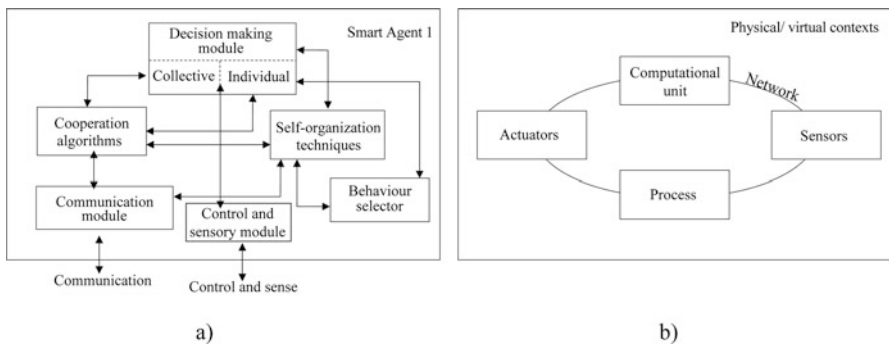


Fig. 3 (a) A proposed operational architecture for Holonic Smart Agent [30, 35]; (b) General CPS architecture [33, 34]

their future operation. In order to ensure intelligent operation, smart machines also need to be capable of learning. This learning feature requires data and the information generated from this data, which are collected by detection. A smart system will not only use the data of its own detection, but it will also leverage on co-operation as a multi-component system. Therefore, a device or machine which is capable of co-operation can be defined as a smart device. In other words, different levels of smartness can be distinguished even in case of machines. As an illustration, Fig. 3 shows the architecture of a smart agent. Regarding resilience related aspects, the aforementioned paradigm of pro-active dependability also refers to smart agents that warn other cooperative agents about possible dangers, exactly like humans working together warn each other when they realize other peers are in danger.

3.2 Smart Systems – Smart Multi-agent Systems

Why should individual agents be connected? How could they be connected? There are several reasons for this: autonomy versus team work, strength in unity, self-defence or error-tolerant group coherence.

According to Csermely [36], humans are communal beings, and as a result, our brain has developed in a way that it can list, revise and activate our relationships. The network of relationships is the key to human survival. Social-psychological surveys show that humans divide the world into groups of 5, 15, 35, 80 and 150 individuals, and these groups represent our family, close friends, further acquaintances, and other smaller groups. We are closed into our cognitive space determined by our cognitive characteristics, and it is difficult for us to think and work in a bigger perspective. We can be successful and we can communicate and exist in this small world restricted by our cognitive characteristics. Random networks, where relationships between the elements have been made by chance, and these relationships are easy to make, also have these small-world characteristics. It is easy to make relationships, because neighbours know each other. In these networks, clustering is fast and frequently occurring [36].

The smart operation of machines is essentially in the realisation of the natural analogue. That is the reason for working with self-organising elements. It is easy to see that one agent is not enough to perform a complicated task. Therefore, it is necessary to build a smart multi-agent system of several co-operating agents.

Intelligence depends on the learning capacity of machine systems, which is based on the finding, using, processing, connecting and fusing of data, or, in other words, on generating information. Knowledge Discovery and Data Mining is a research field which aims to create knowledge from this immense amount of data. This can be automated by using smart agents. With the help of these co-operative systems, which are capable of learning, a complex adaptive system can be built. In nature, for example, in case of floods, fire ants are able to link their bodies together and float on the surface of water saving their own lives and their colony. A swarm intelligent system can also be made by the co-operative operation

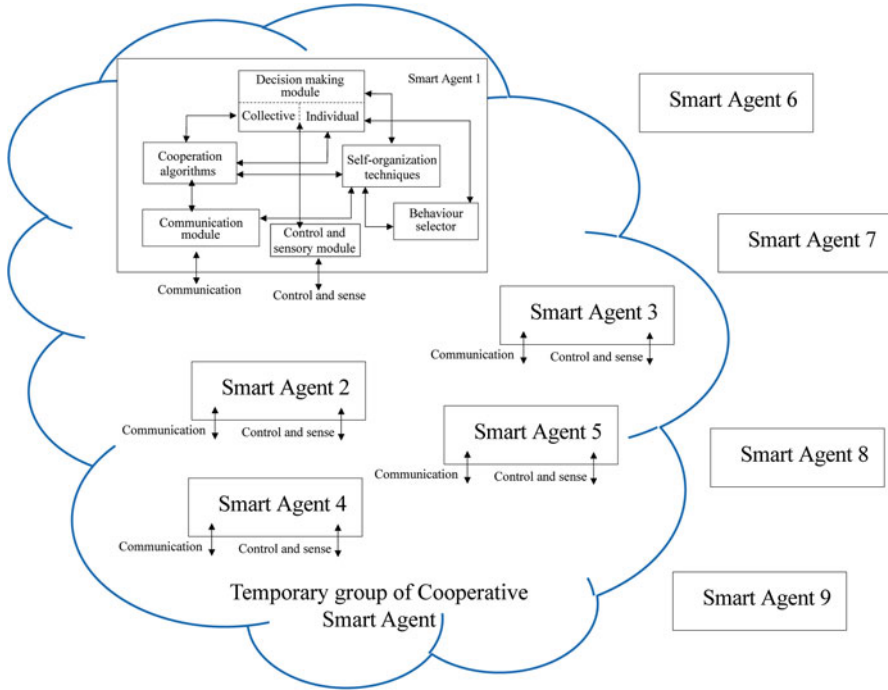


Fig. 4 Smart multi-agent system architecture – random temporary cluster

of machines, and this system can be recognised by the interactions between the above-mentioned smart agents and the agents in their immediate environment or between the agents and the environment. In this system, cognitive science can help to realise machine learning. “Complex systems network theory provides techniques for analysing structure in a system of interacting agents, represented as a network”. (see Fig. 4) [37].

3.3 *Smart Complex Systems Network – Ad-hoc Networked Smart Multi-agent System Sociograms*

Following the analogy with nature, the example of cells can be mentioned in the way they work and perform their tasks in the human body. A complex artificial world can be created by computers and the programs running in their memories, as today there is a tiny computer in almost every device. In order to perform their tasks, machines need an executable program. The more precisely we would like to instruct machines, the more detailed programs are needed, and this can be expensive, time-consuming and requiring considerable resources. The solution to this problem lies in

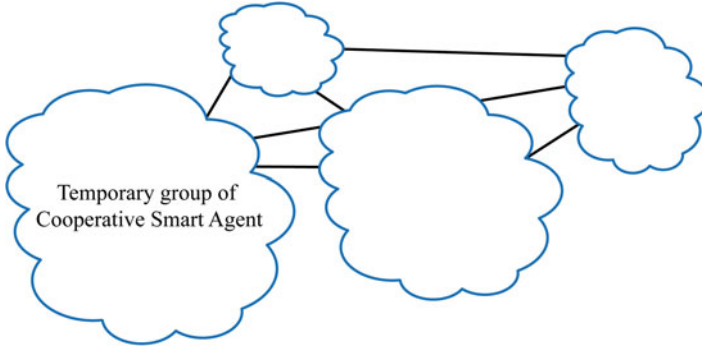


Fig. 5 Ad-hoc networked smart multi-agent system sociograms

the autonomy of machine systems. By creating a network in which smart agents are able to autonomously co-operate, coordinate and communicate with each other for a common purpose – a temporarily-organised network to perform specific tasks – and to make any related decisions and interventions (actions), an ad-hoc networked Smart Multi-Agent System could be built (see Fig. 5).

The reason for organising such networks or clusters can be, for example, that in case of single smart agents, the embedded functions are not always sufficient to perform the given tasks. Furthermore, they can only address the challenges of today’s fast-changing world promptly and find the solution to a special problem if they work together. Regarding resilience, that also holds when smart-agents need to counteract threats or restore from failures requiring a joint effort of multiple cooperating agents.

4 From Machine Intelligence Theory to Smart Machine Theory

According to Bein et al. [38] Machine Intelligence Quotient (MIQ) consists of four key attributes, such as Autonomy, Human–Machine Interaction, Controllability for Complicated Dynamics and Bio-inspired Behavior. The Machine Intelligence Quotient is a measure to assess the intelligence of a Machine. Each attribute has a number of major components. In case of Autonomy, these components are self-calibration, self-diagnostics, self-tuning and fault-tolerance. Human–Machine Interaction includes human-like understanding communication, emergence of artificial emotion and ergonomic design, while Controllability for Complicated Dynamics covers non-conventional, model-based, adaptation, motion planning and non-linearity. Bio-inspired Behavior involves Neuro-science, Biologically motivated behavior, Cognitive-based. According to the theory, in the model space Autonomy and Human–Machine Interaction are constant factors, while Controllability for

Complicated Dynamics and Bio-inspired Behavior are application specific. It is important to define the conditions of use for the model environment, which is done according to the original theory of dynamic, unstructured, and uncertain characteristics of Environment. It suggests three methods to determine MIQ: Fuzzy Logic, Neural Networks and Genetic Algorithms (e.g. neuro-fuzzy-genetic controller for robot manipulators [39]). According to Bein et al., any intelligent system with those features can lead to improved safety, enhanced reliability, higher efficiency and sustainability [38].

4.1 Smart Cyberspace Theory – The Intelligent Cyberspace and the Smart Cyberspace

Figure 6 shows those spaces which are created by human activities and thinking. In the centre of the space, the human can be found, as an individual who is physically present only in real space, but whose mind creates the virtual space, and can extend his physical activities into this virtual space. Physical space represents the geosphere of the Earth. Biosphere is only a part of this space. This is the biological space where the conditions of life are ensured for humans and other living creatures on Earth. Anthropogenic space refers to the world created by humans. Cities, infrastructures and man-made facilities can be found in this space. Individual space makes only a small part of anthropogenic space use by the individual. This is where the individual lives and moves, and where, by his personal activities, he creates the smallest space surrounding him. With regards to humans, the above-mentioned spaces depend on the development level of the society, or, among others, the age, the income, etc. of the individual. For example, the individual space is much smaller for a child than for a professor at the top of his career or for a constantly travelling businessman who is able to influence a more extended space [40].

Virtual spaces are built from the real space and they are shaped by the thoughts of the individual (cognitive space). This cognitive space includes the individual's view of the world, which is the mind's mapping of the physical world in a subjective way. Cyberspace is the artificial world of man-made devices, systems and networks beyond the physical space. A typical example of this is the cyber world of the Internet. Cyberspace is the mapping of the physical space, for example, in case of the Internet, the servers, optical cables, routers and nodes of which the real physical infrastructure is made. Cyberspace inherits the flexibility of the cognitive space, therefore its construction, transformation or use may only be limited by human imagination. Fictional space stands the furthest from reality in comparison with other virtual spaces. At the same time, fictional space can also include real elements. Quite obviously, these imaginary spaces are not in the scope of this study. A smart agent is capable of the interactions shown in Fig. 6 [40].

From the viewpoint of this study, there is a growing number of applications where the two MIQ factors (Controllability for Complicated Dynamics, and Bio-inspired

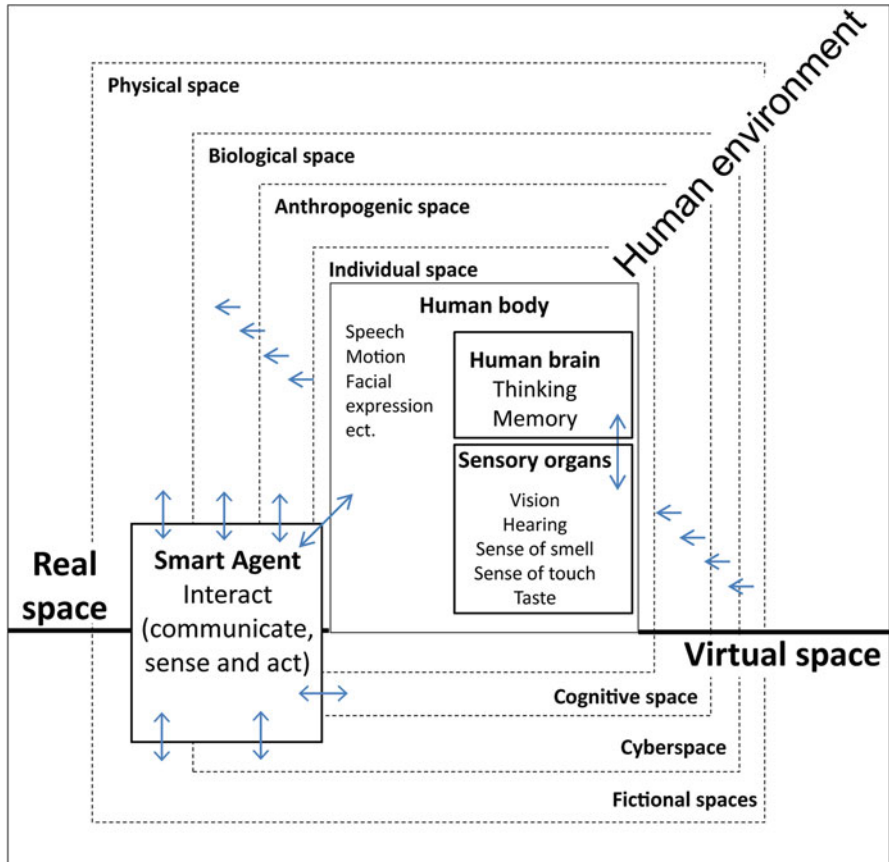


Fig. 6 A model of human perception versus human-like machine perception [35, 40]

Behavior) have equal importance. Therefore, we created an improved version of the above model, and we can state that there are certain cases when these two factors co-exist in space and time.

The geometry of more than three space dimensions can be mathematically described. The fourth space dimension is perpendicular to the other three (X, Y, Z) dimensions, which are commonly used in everyday life. The space defined by four dimensions (X, Y, Z, W) is called a four-dimensional space. The fourth dimension is represented by the W axis. This concept can be difficult to demonstrate in three dimensions, but Fig. 7 shows a generally accepted illustration of a four-dimensional space. The Intelligent Cyberspace can be imagined with the help of the four-dimensional space theory. This space, which is illustrated by Fig. 7 as a hypercube, contains the Machine Intelligence Index (marked in red in Fig. 7) defined by the four attributes/dimensions which characterise Machines. For each device, this point can be found in a different part of space depending on the values of the attributes.

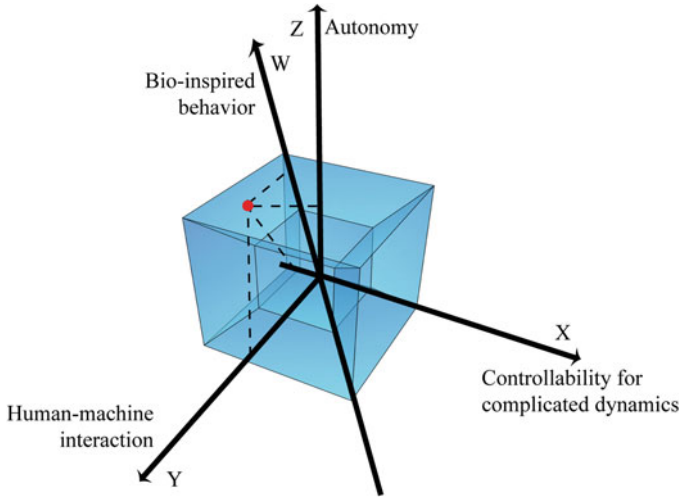


Fig. 7 Forth dimensions as the intelligent cyberspace, visualization of machine intelligence index [own]

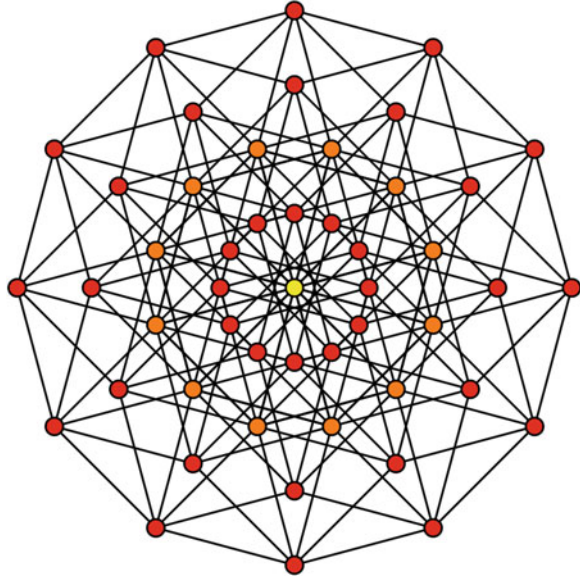
The major components of the main attributes have already been defined, but after 15 years we felt it was necessary to redefine them. Table 1 contains these revised definitions.

It is quite difficult to illustrate a four-dimensional space in two dimensions. In case of a six-dimensional space, however, it is almost impossible. Figure 8 shows one way of representing a six-dimensional space in two dimensions. The Smartness Quotient will be defined in this space.

The Smartness Quotient can be illustrated with the help of the Hexeract. The six dimensions are the following: Dim. 1: Autonomy, Dim. 2: Human–Machine Interaction, Dim. 3: Controllability For Complicated Dynamics, Dim. 4: Bio-Inspired Behavior, Dim. 5: Usability Index Of Machine, Dim. 6: Environmental Performance Index of Machine. The values of these dimensions determine a point in the special six-dimensional Smart Cyberspace. The six-dimensional hypercube represents the space in which the Smartness Quotient is a point, like in the case of the MIQ.

Why should spaces be discussed in such detail? The world and its higher dimensions have always interested humans, and it could not be possible to imagine the existence of smart devices if we did not know how they can become smart. The above described theory can show us how human mind can create from smart machines and devices a space that is no longer cognitive or fictive, but a real Smart Cyberspace.

Fig. 8 Hexeract [41]
 (six-dimensional
 hypercube) – Petrie polygon
 orthographic projections –
 The representation of the
 sixth dimension as the smart
 cyberspace



5 Smartness Theory, Smartness Quotient

What does Smartness Quotient mean exactly, what is it used for and what does it describe? It is the measure of the success factor of the people-centred planning, implementation, use, operation, disassembling and material recycling of machines in relation to environmental effects and from the point of user experience. Its use is based on the view of holistic systems, from the planning, through operation to disassembling and material recycling of the machines. It shows how efficiently a machine can work as an artificial form of life in interaction with humans, the environment and with other machines. With the help of Smartness Quotient, it is possible to compare machines in an objectively quantitative way, in terms of how they are capable of ensuring a safe, sustainable, efficient and convenient life for humans. It can also help to determine the direction of machine development and to define Machine Service Quality on the basis of empiric research. In case of traditional MIQ, the operation of “simple” machines can be examined in a four-dimensional space with a functionality-based approach, while SQ ensures a knowledge and ability-based examination in a six-dimensional space, which supports the development of machines with artificial intelligence with the user in focus.

6 Conclusion

This chapter has presented the basics of developing smart machines in a micro and macro perspective, from machine intelligence, through the agents capable of machine learning, to the theory of networks. Such a view is based on a multi-dimensional space including MIQ and SQ attributes. We have pointed out the difficulties of the path leading from machines to smart machines, and discussed the previously defined and designed characteristics and major components which turn a machine into an artificial form of life, in the way Albertus Magnus had imagined.

One of the further objectives of research is to make the SQ value of machines calculable by quantifying the main components in order to measure and categorise the beneficiary effects to the society. Besides human failure analysis and usability planning, further research will need to deal with hidden errors in technical systems and automatic corrections by smart-troubleshooting and self-healing. A further refinement of the major components of EPIoM is possible by defining the parameters strongly related to smart machines for the more general components.

The development of smart systems is part of the endeavour to ensure human development, survival and well-being. Machine intelligence and smart systems are transforming digital societies. In the Internet of Things era, smart-agents embedded in cyber-physical systems enable new paradigms for Intelligent Transport Systems, Smart Cities and Smart Factories, where the complexity of infrastructural networks and its components is growing exponentially. The use of machine intelligence will also result in increased resilience (including reliability, safety, security, maintainability, self-healing, etc.), efficiency and sustainability in future generation cyber-physical systems.

References

1. National Institute of Standards and Technology (2017) Cyber-physical systems. [Online]. Available: <https://www.nist.gov/el/cyber-physical-systems>. Accessed 31 Dec 2017
2. Farid AM (2015) Designing multi-agent systems for resilient engineering systems. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), vol 9266, pp 3–8
3. Iantovics LB, Gligor A, Georgieva V (2017) Detecting outlier intelligence in the behavior of intelligent coalitions of agents. In: 2017 IEEE congress on evolutionary computation (CEC), pp 241–248
4. Park H-J, Kim BK, Lim KY (2001) Measuring the machine intelligence quotient (MIQ) of human-machine cooperative systems. *IEEE Trans Syst Man Cybern – Part A Syst Humans* 31(2):89–96
5. Park HJ, Kim BK, Lim GY (1999) Measuring machine intelligence for human-machine cooperative systems using intelligence task graph. In: Proceedings 1999 IEEE/RSJ international conference on intelligent robots and systems. Human and environment friendly robots with high intelligence and emotional quotients (Cat. No.99CH36289), vol 2, pp 689–694
6. Ozkul T (2009) Cost-benefit analyses of man-machine cooperative systems by assesment of machine intelligence quotient (MIQ) gain. In: 2009 6th international symposium on mechatronics and its applications, pp 1–6

7. Li C, Ji Z, Pang Z, Chu S, Jin Y, Tong J, Xu H, Chen Y (2016) On usability evaluation of human – machine interactive Interface based on eye movement. In: Long S, Dhillon BS (eds) *Man-machine-environment system engineering: proceedings of the 16th international conference on MMESE*. Springer, Singapore, pp 347–354
8. Szabó G (2018) Usability of machinery. In: Arezes P (ed) *Advances in safety management and human factors: proceedings of the AHFE 2017 international conference on safety management and human factors*, July 17–21, 2017, The Westin Bonaventure Hotel, Los Angeles, California, USA. Springer International Publishing, Cham, pp 161–168
9. Aykin N (ed) (2005) *Usability and internationalization of information technology*. Lawrence Erlbaum Associates, Inc., Publishers, Mahwah
10. Hsu A et al (2016) Global metrics for the environment. In: *The environmental performance index ranks countries' performance on high-priority environmental issues*. Yale University, New Haven
11. Liouane Z, Lemlouma T, Roose P, Weis F, Liouane Z, Lemlouma T, Roose P, Weis F, Neu HMAG (2017) A genetic neural network approach for unusual behavior prediction in smart home. In: Madureira AM, Abraham A, Gamboa D, Novais P (eds) *Advances in intelligent systems and computing*, vol 2016. Springer International Publishing AG, Porto, pp 738–748
12. Hollós J (1979) *Az ipari elektronika alapfogalmai*, 3. Műszaki Könyvkiadó, Budapest
13. Katsnelson A (2012) DNA robot could kill cancer cells. *Nature*. <https://doi.org/10.1038/nature.2012.10047> <https://www.nature.com/news/dna-robot-could-kill-cancer-cells-1.10047>
14. Cosic J, Curkovic P, Kasac J, Stepanic J (2013) Interpreting development of unmanned aerial vehicles using systems thinking. *Interdiscip Descr Complex Syst* 11(1):143–152
15. Mester G, Plet S, Pajor G, Jeges Z (1992) Flexible planetary gear drives in robotics. In: *Proceedings of the 1992 international conference on industrial electronics, control, instrumentation, and automation*, vol 2, pp 646–649
16. Wahlster W (2013) *The semantic product memory: an interactive black box for smart objects*. Springer, Berlin/Heidelberg, pp 3–21
17. Mester G, Aleksandar R (2010) Sensor-based intelligent mobile robot navigation in unknown environments. *Int J Electr Comput Eng Syst* 1(2):1–8
18. Tokody D (2018) Digitising the European industry – holonic systems approach. *Procedia Manuf* 22:1015–1022
19. Rouse M. Smart machines. [Online]. Available: <https://searchcio.techtarget.com/definition/smart-machines>. Accessed 28 Apr 2018
20. Crowcroft J (2011) *System design: an engineering approach to computer networking*. Cambridge University, Cambridge
21. Lee J, Bagheri B, Kao HA (2015) A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manuf Lett* 3:18–23
22. Tokody D, Flammini F (2017) The intelligent railway system theory. *Int Transp Spec Ed Int Verkehrswesen* 69(1):38–40, ISSN 0020–9511
23. Øien K, Jovanović A, Grøtten TO, Choudhary A, Øren A, Tetlak K, Bodsberg L, Jelic M (2017) Assessing resilience of smart critical infrastructures smart resilience based on indicators. *Zenodo*, Stuttgart, p 87
24. Jovanovi A, Choudhary KØ, Choudhary A (2018) An indicator-based approach to assessing resilience of smart critical infrastructures. In: Fekete A, Fiedrich F (eds) *Urban disaster resilience and security*. Springer International Publishing, Cham, pp 285–311
25. Schank R. Chapter 8: Smart machines. [Online]. Available: https://www.edge.org/conversation/marvin_minsky-chapter-8-smart-machines. Accessed 21 Aug 2017
26. Béky Z (2013) *Az orvosi eszközök használhatósági tervezése (Usability Engineering)*. B. Braun Medical Kft
27. Electrotechnical Commission International (2007) IEC 62366: 2007, medical devices — application of usability engineering to medical devices. International Electrotechnical Commission
28. Wang Z, Zhang B, Guan D (2016) Take responsibility for electronic-waste disposal. *Nature* 536(7614):23–25

29. Necula D, Vasile N, Stan MF (2013) The impact of the electrical machines on the environment. In: 2013 8th international symposium on advanced topics in electrical engineering (ATEE), pp 1–4
30. Tokody D, Flammini F (2017) Smart systems for the protection of individuals. *Key Eng Mater* 755:190–197
31. Iantovics LB, Rotar C, Nechita E (2016) A novel robust metric for comparing the intelligence of two cooperative multiagent systems. *Procedia Comput Sci* 96:637–644
32. Iantovics BL, Crainicu B (2008) Complex mobile multiagent systems. In: 2008 first international conference on complexity and intelligence of the artificial and natural complex systems. Medical applications of the complex systems. *Biomedical Computing*, pp 21–30
33. Esfahani A, Mantas G, Yang D, Nascimento A, Rodriguez J, Neves J (2015) Towards secure network coding-enabled wireless sensor networks in cyber-physical systems. In: *Cyber-physical systems*. CRC Press, pp 395–414
34. Zanni A (2015) Cyber-physical systems and smart cities learn how smart devices, sensors, and actuators are advancing internet of things implementations. *IBM developer works*, no. April, pp 1–8
35. Velik R (2008) A bionic model for human-like machine perception. Vienna University of Technology
36. Csermely P (2009) Why do we like networks? In: *Weak links: the universal key to the stability of networks and complex systems*. Springer, Berlin/Heidelberg, pp 5–52
37. Crowcroft J (2011) *Network/graph network/graph theory theory*. University of Cambridge
38. Bien Z, Bang W-C, Kim D-Y, Han J-S (2002) Machine intelligence quotient: its measurements and applications. *Fuzzy Sets Syst* 127(1):3–16
39. Mester G (1995) Neuro-fuzzy-genetic controller design for robot manipulators. In: *Industrial electronics, control, and instrumentation, 1995. Proceedings of the 1995 IEEE IECON 21st international conference on, 1995, vol 1*, pp 87–92
40. Pirisi G, Trócsányi A (2011) *Általános társadalom – és gazdaságföldrajz*. Pécsi Tudományegyetem, Pécs
41. 6-cube. [Online]. Available: <https://en.wikipedia.org/wiki/6-cube>. Accessed 21 Aug 2017
42. Wechsler D (1940) Non-intellective factors in general intelligence. *Psychol Bull* 37:444–445

Challenges and Opportunities for Model-Based Security Risk Assessment of Cyber-Physical Systems



Marco Rocchetto, Alberto Ferrari, and Valerio Senni

Abstract The design of Cyber-Physical Systems (CPS) poses a number of challenges, in particular for cyber-security. Eliciting Security Requirements is a key aspect in the early system design stages; however it is important to assess which requirements are more stringent and grant protection against the higher-value assets. Cyber-security Risk Assessment (SecRA) has a key role in determining threat scenarios and evaluating the risks associated to them but it is a practice that has been principally developed for IT systems, thus focusing on cyber threats. In this chapter, we discuss the state of the art in SecRA methodologies and the challenges to be addressed for developing new CPS-oriented SecRA methodologies. Based on the most relevant standards for industrial control systems and automotive domain (such as the ISA/IEC-62443 and the J3061), we propose the adoption of an asset-driven viewpoint and a model-based approach to SecRA, and we identify current gaps. In particular we discuss (i) CPS (security) modeling languages and methodologies, (ii) vulnerabilities cost models and the network of public repositories of vulnerabilities, (iii) attacker models and profiles, and (iv) complex cyber-physical attack chains. Finally, we discuss our vision, focusing on assets and leveraging model-based design practices can provide a more rigorous approach to SecRA for CPS, allow taking into consideration their peculiarities, and support to manage the large complexity involved in their operation. The desired outcome is to provide the system design team with methods and tools to identify complex attacks and perform a cost/benefit tradeoff analysis to justify the adoption of specific Security Requirements and the necessary costs implied by the corresponding mitigations.

M. Rocchetto · A. Ferrari · V. Senni (✉)
United Technology Research Center, East Hartford, CT, USA
e-mail: valerio.senni@utrc.utc.com

© Springer International Publishing AG, part of Springer Nature 2019
F. Flammini (ed.), *Resilience of Cyber-Physical Systems*,
Advanced Sciences and Technologies for Security Applications,
https://doi.org/10.1007/978-3-319-95597-1_2

1 Introduction

The design of modern cyber-physical, embedded systems poses complex challenges in terms of cyber-security. The adoption of (1) more complex and distributed electronics (e.g. in the automotive domain [1]), (2) an increasing number of software components (e.g. as reported in [2] the Airbus A320 avionics system has around 80,000 lines of code while the Boeing 777 exceeds 4 million lines), and (3) remote monitoring, configuration, maintenance, and software update capabilities (e.g. firmware update over-the-air [3]) provide new entry-points to security threats. Therefore, system designers need to face the complexity of securing a significantly increased system attack surface. Following a secure-by-design approach [4], several standards provide guidelines and processes to support the design and deployment of secure systems by careful evaluation of external system entry-points and internal architecture from the perspective of an attacker (e.g. ISA/IEC-62443 for Industrial Control Systems [5], the ISO/IEC-27000 family [6] and NIST 800-53 for IT Security [7], DO-356 for Avionics Industry [8], and J3061 for the automotive domain [9]). A key step, identified in the processes proposed by standards, is that of *Cyber-security Risk Assessment (SecRA)*, which supports the identification of threats and the evaluation of the risks to which the system is exposed. The principal outcome of SecRA is a set of *mitigations* for the higher criticality risks, which are translated into *derived security requirements* and added to a set of Minimum Security Requirements (MSR) either (1) obtained directly from the customer or (2) product/domain-specific, or (3) inherited from regulations and standards (such as the common criteria of ISO/IEC-15408 [10]). See Fig. 1 for a conceptual workflow showing the role of Risk Assessment and compare also to the J3061 process [9].

SecRA is typically performed as a structured process [11] that guides through the identification of the key system *assets*, the elicitation of relevant *threats* and of the vulnerabilities they can leverage, and an informal evaluation of the *impact* (with an

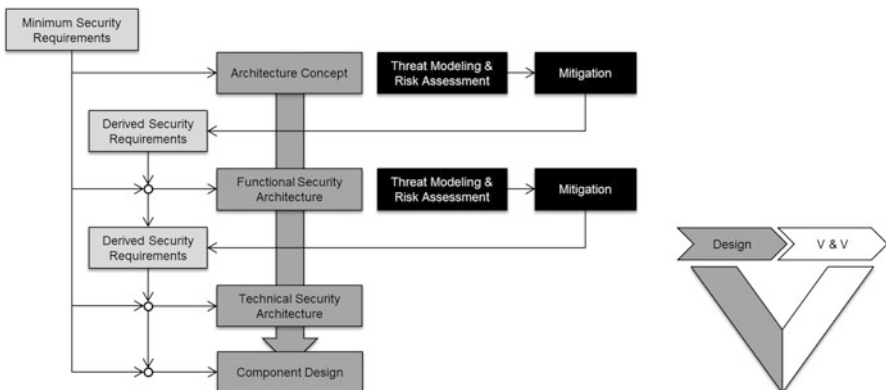


Fig. 1 Role of risk assessment in a secure-by-design workflow

associated *likelihood*) that those threats can have on the system assets. Knowledge of both the application domain and of the (expected) system design is critical to ensure an effective evaluation of the risks.

In this chapter, we discuss two potential shortcomings of this common approach to SecRA. First, the reasoning behind the analysis of impact and likelihood is often not supported by formal artifacts (e.g. functional/architectural diagrams [12, 13] or attack trees [14]). Thus, the evaluation of *completeness of the assessment and correctness of the results* relies entirely on the system engineers (and/or security experts), with the side effect of producing *few or no artifacts that document the rationale behind the analysis* (e.g., for future and third-party reviews). Second, the classic SecRA process provides limited support for the *analysis of the impact on risk evaluations of any system design change*, due to the absence of the aforementioned architectural artifacts and consequent lack of traceability of risks to system design elements (e.g. to architecture elements).

The role of SecRA is well understood for IT/software systems (and a number of tools exist to support it [15]), but this is not the case for Cyber-Physical Systems (CPS), which are complex systems involving software, hardware, actuators, sensors, plant and environment interactions with strict performance and safety constraints [16]. The challenge of secure design of these systems stems from the complex interaction between physical and virtual components, exposing the physical world to the effects of cyber threats, with direct impacts on safety (e.g., Stuxnet [17] and other examples in [18]). Therefore, the aforementioned shortcomings of common SecRA approaches are more relevant for CPS and cannot be overlooked.

Our *first claim* is that we envision a fruitful synergy between Model-Based Design [19] and SecRA resulting in an “MB-SecRA” approach that can (1) improve confidence on completeness and correctness of SecRA results by providing reviewable artifacts documenting the risk assessment rationale, and (2) provide support to change management by explicit traceability of risks to design elements.

A second challenge, rising in modern CPS, is the increasing complexity of hardware and software, which opens CPS to the risk of sophisticated *multi-step attacks*. A notable example is the aerospace domain; with a tremendous increase in use of software to replace hardware functionalities (e.g. see the study on software complexity from NASA [20]). Similarly, in the automotive domain, the number of lines of code (LOC) and of CPUs present in a modern car makes the system highly subject to advanced cyber-attacks [9]. In commercial applications, to reduce the time to market, it is common to reuse existing software components, protocols or platforms and this raises the concern of how wide is the impact of a known vulnerability. There are large public databases such as OWASP [21] or CWE [22] that report hundreds of vulnerabilities affecting widely adopted software components. For hardware components there are no databases of equal scope yet but we envision a growing need in this area. This complexity of the software/hardware architecture opens up for complex attacks known as *advanced persistent threats* (APT), where an attacker (willing to invest resources and time to archive its objectives) is able to reach core system assets by exploiting multiple (apparently not so critical) vulnerabilities in a synergistic way (*kill- or attack-chains* [23]) to

achieve increasingly more information about the system and higher privileges, until he is able to produce system-level damages. Kill chains and APTs are very hard to identify at system design time and also at run-time due to their sophistication [24].

For a correct evaluation of cyber-security risks in complex systems it is necessary to change the traditional approach of considering the impact of a single vulnerability in isolation and turn the attention to attack chains. From this perspective, a vulnerability may be discovered to play a critical role in multiple attack chains and thus become one of the highest-ranked among the identified mitigation actions.

A growing number of Security Risk Assessment tools and methodologies (e.g., CORAS [25], ThreatModeler [26], Microsoft STRIDE [27]) adopt design artifacts such as software/hardware architecture schemes, network topology diagrams, and data-flow diagrams to support the evaluation of system-level effects of local vulnerabilities. However, they leave the actual evaluation of the impact and the risk caused by those vulnerabilities to a manual and informal analysis of those artifacts performed by a system engineer. This direction seems promising but still suffers from two issues: first, *the number of vulnerabilities to be combined can give rise to hundreds of potential attack scenarios* and a high review complexity, and second, there is *limited documentation of the reasoning behind the analysis of system-level impacts of local vulnerabilities*. Our second claim is that, the contribution of MBD into Cyber-security Risk Assessment should not be limited to driving the identification and evaluation of risks (e.g., by means of formal architectural artifacts) but also leverage abstract behavioral models that allow the representation of data- and function-flows that an attacker can use to propagate the effects of a vulnerability exploitation. A successful approach in the design of high-assurance systems has been the adoption of formal, automated and exhaustive analysis methodologies to ensure the absence of undesired behaviors in software design (e.g. see adoption of formal methods in avionics [28]), including absence of cyber-security vulnerabilities such as potential attacks in protocols design [29]. We believe formal security analysis can be successfully applied also in the area of Risk Assessment.

The objectives of this chapter are (1) to identify challenges and opportunities to improve current SecRA methodologies for the specifics of CPS, and lay the basis of an MB-SecRA approach to improve confidence on completeness and correctness of risk assessment by leveraging formal and traceable model-based artifacts and (2) provide a high-level a workflow for formal analysis of known vulnerabilities and identification of attack-chains, and finally, (3) to discuss several open challenges and gaps that need to be filled to realize the proposed MB-SecRA approach.

1.1 Structure

In Sect. 2, we discuss the state-of-the-art of SecRA and the open challenges for CPS. In Sect. 3, we discuss the opportunities we envision in developing a model-based approach to system, attacker and vulnerabilities modeling for Risk Assessment. In Sect. 4, we describe a possible roadmap to implement model-based SecRA and use a small example to illustrate the concepts.

2 Background and Open Challenges

Cyber-security Risk Assessment has been extensively discussed over the past years [30, 31] and attracted special attention in the field of SCADA systems [32] given their role in managing and controlling critical infrastructures. Despite the large amount of work in the field, this area is in practice still widely addressed by using informal artifacts and tools [33] (such as Excel spreadsheets) and strongly relies on the domain expertise of review teams.

In this section, we discuss the open challenges that we identified for the application of model-based SecRA for CPS. Moreover, we discuss the opportunities to increase the effectiveness of SecRA in terms of (1) completeness and correctness of the results, (2) capability of managing design changes, and (3) understanding of system-level effects of attack-chains.

2.1 *Cyber-Security Risk Assessment Methodologies*

Following to the ISO 31000 [31] (a consolidated standard providing a framework for risk management), Risk Assessment is characterized by three main activities: (1) Risk Identification, (2) Risk Analysis, and (3) Risk Evaluation.

In the cyber-security context [30], Risk Identification is the process of recognizing and describing risks. Based on the *Assets* (what is protected), the *Incidents* (events that have negative consequences on the Assets), and the system cyber-security *Vulnerabilities* (design or implementation flaws that, if exploited, can cause an Incident), a *Risk* identifies the conditions under which external or internal *Threats* can exploit existing vulnerabilities with the purpose of causing an incident and, thus, a damage to the Assets. The Risk Analysis activity has the objective to review the identified risks and to provide a quantitative estimate for the *likelihood* of a specific risk and the related *impact* on assets. Finally, during the Risk Evaluation activity, each risk is compared with the evaluation criteria. The quantitative estimate of impact and likelihood is used to determine the risks that should be considered for treatment. Risk treatment typically implies the identification of derived security requirements indicating the required security measures, e.g. elimination of vulnerabilities or restriction of accessibility. Several methodologies for SecRA leverage a representation of the system architecture and data-flow to support a more formal and repeatable approach to Risk Identification and Risk Analysis.

Microsoft proposes a methodology based on (1) the threat classification model STRIDE (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privileges) and (2) a language for threat modeling based on abstract software elements (Entities, Processes, Data-flows, Data Stores, Trust Boundaries) to represent and analyze paths that links vulnerabilities to incidents [34]. STRIDE then allows to choose to proceed either “per component” or “per interaction” to perform the assessment of risks. A tool (Microsoft Threat Modeler)

implements STRIDE “per component” and supports the application of STRIDE categories to each component in the threat model. Pre-determined templates are available for typical software components (e.g. services, databases) with specific questions to characterize the risk for the specific component. In this methodology the architecture model has mainly the role of supporting reviews and ensuring completeness of SecRA, but does not allow understanding the actual effect at system-level of the component vulnerabilities. This focus on a specific library of software components together with the lack of system-level assessment leads us to consider this methodology not adequate to CPS.

The SESAR SecRAM methodology [35] has an implicit dependency to the notion of system architecture. It starts from the identification of *Primary Assets*, that are abstract critical system functions or services, and adopts an approach similar to STRIDE by considering a categorization of the threats based on the CIA (Confidentiality, Integrity and Availability) paradigm. Based on these categories, SecRAM is able to provide a preliminary assessment of the impact areas and a corresponding evaluation of criticality. By leveraging a representation of the physical architecture of the system, SecRAM maps the *Primary Assets* to the *Physical Assets*: impacts are inherited in this mapping process. Thus, the methodology can proceed backwards considering the vulnerabilities of each component and producing chains Vulnerability-Impact-Asset. The final step consists in computing the likelihood of the risks. The advantage of SecRAM is to take into account the system-level by starting from *Primary Assets*. The mapping to the (physical) architecture provides a better understanding of how the vulnerabilities affecting single (physical) components might impact on *Primary (Immaterial) Assets*. Still, this valuable information is provided by the system and security engineers through an informal review.

CORAS [25, 36] is an explicitly model-driven SecRA methodology in the sense that models are not implicit but are adopted to support and execute all the Risk Identification, Analysis and Evaluation activities. The diagrams and views provided by CORAS are designed to be straightforward and to enable the capture and documentation of relations between threats, vulnerabilities, incidents and impact on assets. The diagrams are created with the purpose of supporting discussion and documenting *structured brainstorming sessions*. CORAS is the methodology closer to our vision since its process is based on a functional justification of the vulnerabilities effects. Threat Diagrams are graphs structured into specific layers: (1) a mapping of Assets to Incidents affecting them, (2) a mapping of Incidents to a chain of internal System States and Vulnerabilities enabling this chain, and (3) a mapping of vulnerabilities to Threats that can exploit them. This representation provides a valuable representation of the potential path of an attack (Threat Scenario), it is useful to justify how a Threat agent is expected to exploit Vulnerabilities to affect the Assets, and clarifies the risk analysis rationale. However, Threat Diagrams are very high-level and applicable for the concept design phase [12] (i.e., the very first engineering design step). They are created in a non-rigorous way by system and security engineers and their level of abstraction makes it hard to extract concrete attacks from the results (such as those obtained from CWE

[22], NVD [37] or threat reports). The motivation is to be found in the fact that Threat Diagrams are unrelated to the actual system logical/physical architecture and behavior. Thus, there is no formal justification for the identified Threat Scenarios, captured and described by the analyst based on his experience.

Our claim is that the manual extraction of Threat Scenarios is extremely complex and highly error prone when considering a CPS, and may be unfeasible for large-scale CPS, where there can be thousands of complex attacks. For this reason we see the opportunity of improving state-of-the-art risk assessment approaches to allow the systematic extraction of Threat Scenarios by leveraging different system viewpoints (e.g., logical and physical architectural views, behavioral and vulnerability models, attacker models). Models can also support automated extraction of Threat Scenarios, thus making the Risk Assessment activity less error prone. In the context of model-based system design flows [19], systematic extraction of Threat Scenarios from models can pave the way to the identification of potential attacks that are easier to reproduce on the actual system in the security validation phase.

2.2 CPS Design Languages for Security

The systematic or automated extraction of Threat Scenarios and the consequential analysis of risks are applicable under the assumption that the CPS design is based on models and a specific security viewpoint is captured and documented. In this section, we briefly review the state-of-the-art of system modeling languages and their support for security viewpoints.

CORAS is based on UML [13], a robust and highly adopted [38] (open) standard modeling language for software engineering. In UML, it is possible to define two main types of diagrams: structural (e.g., class and package diagrams), and behavioral (e.g., sequence and activity diagrams). The *structural diagrams* are used to decompose software into different parts (e.g., packages, classes, methods) while the *behavioral diagrams* supports the design of the semantics of those different parts. UML supports mechanisms, called profiles, which allow its users to make extensions to the language itself (i.e., semantics refinements of UML). The CORAS language for risk modeling was initially defined as a UML profile [39, 40], and standardized as part of the UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms (UML QoS&FT). There exist several other profiles that extend UML to model security properties or risk-related information. Examples of the former are SecureUML [41] and UMLSec [42], while examples of the latter are Abuse-Cases by McDermott and Fox [43].

In [43], McDermott and Fox first proposed the idea of applying specialized use-cases for the purpose of threat identification, and misuse-cases [44] (that extends UML use-cases) to elicit security requirements. Another approach is provided by UMLSec, that extends UML with tags (called stereotypes) that allow the modeler to define security properties and constraints. Similarly, SecureUML extends UML allowing the modeler to express access control policies as constraints. The focus of

UML is mainly on software development rather than embedded systems or CPS. On the other hand, SysML [45] (Systems Modeling Language) extends UML to support the design of complex hardware/software systems by introducing new diagrams to define requirements and constraints on the system properties (e.g., performance or reliability) and block diagrams to better describe the structure of hardware/software components and interfaces. Similarly to UML, SysML extensions have been proposed with the goal of introducing security concepts to the SysML language. SysML-Sec [46, 47], developed in the context of the EU project EVITA [48], is one such extension. SysML-Sec is designed to take into account security and safety during the engineering development of embedded systems and shares some commonalities with our vision. Specifically, SysML-Sec leverages SysML block and state machine diagrams enriched with security aspects that allow the modeler to consider both architectural and behavioral aspects. However, SysML-Sec lacks of any physical aspect in attack modeling and no support for considering attack-chains exploiting multiple vulnerabilities in sequence.

The Architecture Analysis and Design Language (AADL) is designed for the specification, analysis, automated integration and code-generation of real-time distributed systems [49]. In [50], the authors consider a set of security requirements that have been proved to mitigate set of attacks (exploiting known weaknesses in the software architecture designs) to authentication and input validation methods. The authors were able to define architecture design constraints that, if satisfied, ensure the satisfaction of those highly relevant security requirements. Exploiting this result and the available analysis tools for AADL they were able to formally prove whether an architecture is robust against attacks to authentication and input validation mechanisms. The approach leverages STRIDE to derive specific authentication and validation requirements for the system under analysis, then these requirements are used to validate the architecture against the above mentioned security requirements. This approach provides high assurance against a fixed set of security requirements that can be expressed in terms of architectural constraints. The challenge we propose is to develop an approach that is not specific for a set of security requirements but can be applied to general security requirements to validate both the system architecture and the system behavior.

2.3 Open Challenges

In this section, after the review and assessment of the state-of-the-art, we summarize the challenges identified for Cyber-security Risk Assessment of CPS:

1. *Model-driven method for Cyber-security Risk Assessment of CPS.* The adoption of models can (a) increase confidence and completeness in risk assessment, (b) provide formal support for a more objective evaluation and documentation of the risk assessment rationale through reviewable artifacts, (c) support change management through traceability of risks to design elements.

2. *Support for risk assessment of complex cyber-physical attacks.* To overcome the complexity of modern CPS hardware and software, the high number of vulnerabilities, and to be able to evaluate risks of sophisticated multi-step attacks we envision the development of tools that can automate portions of the risk assessment leveraging the models fostered in challenge (1). The research community is well aware of this problem and has been actively working on that over the past few years ([51–54] to name a few).
3. *Formal models of CPS.* Modelling a CPS is challenging because it requires to consider physical aspects (e.g. accessibility) together with software and network aspects. Further understanding of potential attacks can be achieved considering also the system behavior, that is, use cases, data flows and components' role in the protection of the system assets. For this reason it is important to develop a view-based approach to system modeling to allow easier maintenance and review of the different viewpoints.
4. *Formal models of vulnerabilities and attackers.* To have a formal approach to assessing complex risk scenarios it is necessary to capture vulnerabilities and their effects on the nominal system behavior. There is still no broadly accepted library of vulnerabilities even though (as we are going to describe in Sect. 3.3.2) there exists a number of libraries of public vulnerabilities. The problem with existing vulnerability libraries is twofold: (a) representation of vulnerabilities is often informal, and (b) for industrial products it is often impossible to retrieve vulnerability reports. For this reason, a vulnerability library is an important asset for a company and should be devised to be reusable to mitigate maintenance costs. A second important part of the risk assessment model is the attacker, which is required to capture the potential interactions with the system as well as the cost and likelihood of the single events.

3 Model-Driven Cyber-Security Risk Assessment

In this section, we discuss some opportunities we currently identified in the roadmap to address the open challenges discussed so far. We consider, in particular, abstraction levels, expressiveness, and complexity of CPS modeling. We propose an approach to vulnerabilities and attackers modeling and we also consider aspects related to cost models.

3.1 CPS Formal Model and Abstraction Level

The adoption of formal system modeling languages is influenced by *expressiveness*, *usability*, and support by *automated tools*. Expressiveness of a formal language is directly related to the *level of abstraction* of the system model. Modeling the system in high details has the advantage of lowering the gap between the model and

the real system, resulting in more precise analyses, at the cost of maintenance and high skills required to develop the models. The correct tradeoff between complexity and abstraction level is the principal aspect to consider. However, for security there is also another axis to be considered, which is the complexity of defining *correct security models and properties*, see for example the interesting discussion on how to define a system to be secure [55]. To give an example of this, a vulnerability may be publicly available and studied, but the impact of that vulnerability has to be related to the specific system and depends also from the perception of the system owner. For this reason Assets are a key input that should be provided by the system owner for any security analysis.

To evaluate the system-level impact of a vulnerability, a model of a CPS should consider both its *architecture* and its *behavior*. The architecture captures the *topology* and the *interactions* between components; while the behavior defines the *dynamics* and *functionalities* of the CPS. The formal modeling of architectures has a fairly extensive reference literature and several surveys exist on the topic (see, e.g., [56] for the protocols, and [57] for the architecture). Formal models of protocols behavior typically rely on transition system expressing how the information is sent (structure of the packets) and the evolution of the knowledge of the parties involved in the communication. Similarly, the architecture should take into account how the topology allows the exchange of information between subsystems. The behavior of the overall system relies on the behaviors of the various components of the system and, in turn, on the inputs/outputs generated and sent between components.

As discussed in [51], the correct modeling of the physical environment (e.g., the dynamic of the system or the laws of physics) of the CPS plays an important role in the correctness of the modeling and of the results of the security analysis. To mitigate the effort and complexity of modeling the physical environment there are opportunities in automated identification of the dynamic of the system [58], which is still an open research challenge. It is important to consider that faithful physical models may be hard to analyze formally [59] and can benefit from domain specific abstractions leveraging expressive formal languages [60]. Core formal analysis engines (e.g., Z3 [61], nlsat [62], Yices [63], MathSAT [64], CVC4 [65]) have made big steps forward in solving non-linear mathematical models, thus making formal analyses closer to be applicable to CPS, see for example the promising benchmarks recently obtained by the NuXMV model checker [66]. In summary, there are several open scientific challenges in security analysis of CPS and we see opportunities emerging for formal and automated extraction of attack scenarios.

3.2 Attacker Models

In the literature there is an extensive list of (formal or semi-formal) attacker models. One of the most widespread attacker models is the so called Dolev-Yao model (DY) [67]. The DY has been extensively used in the past few decades in a number of security protocol verification tools (e.g., ProVerif [56], MaudeNPA [68]). For the

purpose of security analyses, protocols are modeled as a set of agents exchanging messages over a network. The (DY) attacker is usually assumed to be part of the network and to be able to read as well as modify the messages, performing operations such as encryption and concatenation. Protocol analysis typically applies the ‘perfect security’ assumption, where the attacker cannot break cryptographic but only leverage the protocol logic for the attack.

The model of the attacker for CPS shall be different from the ones considered for protocol verification [53]. Usually, the focus is not on cryptography but more on the control of the network extended with some physical properties (e.g., the physical location of the attacker with respect to the CPS [69]). In fact, the physical part of the CPS allows the attacker to perform a number of attacks which extends to physical interactions with the system (e.g., physical-layer interactions or side-channel attacks). There is still no unified theory of a cyber-physical attacker model but some approaches have proposed their attacker model for CPS.

3.2.1 Profiling the Attacker

We remark that the focus of risk assessment is not on identifying new vulnerabilities but rather to be able to precisely estimate the risk associated to known vulnerabilities. Therefore, we are not interested in an attacker that can show, e.g., new flaws on a CPS but on an attacker that can leverage known vulnerabilities to impact the assets of a CPS.

According to [53] “an *Attacker Model* (together with compatible system models) will ideally fully characterize the possible interactions between the attacker and the system under attack. In particular, the model will define constraints for the attacker (e.g. finite computational resources, no access to shared keys)”. Below we summarize the main characteristics of the attacker (i.e., attacker profile) that can be used as a basis for the definition of a vulnerability model. In the description we stress in italics the metrics we derived from [53].

- *Knowledge*. We consider the worst-case scenario and a *fair* attack surface, where the attacker has an incomplete understanding of the system under attack (*system*). The model specifies sub-system that are not directly accessible from the attacker, but also the knowledge of the attacker about those sub-systems and how they work (e.g. what communication protocols are in place, *credentials*, etc.). The attacker has (partial) visibility over a protocol or the functioning of a component but, in general, cannot directly modify the behavior of components (*source code*). Therefore, we can consider cases where the attacker just waits until the system reaches a specific configuration so that he can perform his attacks. The *offensive* skills of the attacker are strongly connected to the vulnerabilities of the system, i.e., an attacker can interact with the system as a regular user (*physical*), and can exploit attacks that leverage vulnerabilities of the system.
- *Resources*. We consider scenarios where the attacker has physical access to the system, and scenarios where the attacker needs to exploit some vulnerability to

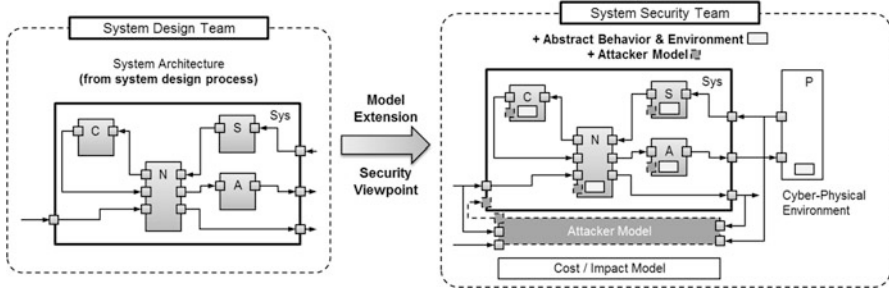


Fig. 2 System design extended with a security viewpoint

access to the system. The *effort* that the attacker puts into his attacks is also an important parameter. It is possible to formalize the notion of cost for each attack step and have an analysis to take into account the cost model for effective analysis of the most likely attack.

- *Psychology*. The attacker is *dishonest* and driven by (i) the maximization of the impact of a vulnerability on the nominal behavior of the system and on his knowledge, and (ii) the cost of exploiting such vulnerability (*aim*). The *strategy* of the attacker is driven by the exploitation cost of the vulnerabilities and the assets of the system model.

The attacker model can be seen as an extra component that is able to stimulate both the nominal inputs of the system and also the vulnerabilities, which are able to change the nominal behavior of the system. We also envision having another element of the model that is able to capture costs and impacts. The attacker may choose to exploit a vulnerability based on the cost associated to the relative attack. In Fig. 2, we provide a conceptual representation of how a system design model can be extended with a model of the attacker and a model of costs/impacts.

The key aspects that we have not discussed yet are how to model vulnerabilities, their effects on the components behavior, and the cost of exploitation.

3.3 Vulnerability and Cost Models

Formal risk assessment requires correct modeling of vulnerabilities and their effects [70]. Typically, public vulnerability repositories do not provide a description of vulnerabilities effects that is sufficiently detailed to support *formal* modeling. This is not surprising, because vulnerabilities effects are software/hardware/OS-dependent and are much better described in terms of the mechanism that they exploit, to allow covering several cases and conveying the logic of the attack, rather than the technical detail. Therefore, a model of the vulnerability effect is an effort that is application-specific and becomes an important asset of a company. To mitigate this effort it is

important to define ad-hoc libraries (which often remains private) [71] that enable reuse. Another approach is to encode vulnerabilities in the system models as it is done, for example, in security mutation testing [72]. In order to provide a general methodology, similarly to [71], we now discuss the opportunities available in public software vulnerability repositories.

The National Vulnerability Database [37] (NVD), is a public database provided by the National Institute of Standards and Technology of the U.S. Department of Commerce. NVD provides a detailed overview of each vulnerability by providing (1) an informal description of the attack vector and the conditions under which the vulnerability is exploitable, (2) affected software and versions, and (3) a scoring number, defined according to the Common Vulnerability Scoring System (CVSS) [73] and providing an evaluation of the attack complexity and of the potential impacts. Every NVD description has a unique Common Vulnerability and Exposures (CVE) identifier and description [74]. The CVE database is maintained by the MITRE organization with the objective of facilitating and standardizing information exchange on vulnerabilities.

The Open Vulnerability and Assessment Language (OVAL) [75] is an important tool for automated assessment of vulnerabilities: it provides a formal description of the necessary preconditions for exploitation of a CVE entry. OVAL descriptions are structured to be processed by an automated tool in order to evaluate whether a CVE is applicable to a specific system/environment.

Another important source of information is the database of Common Weakness Enumerations (CWE) [22]. A CWE entry describes a weakness that can occur in a software architecture, design, code or implementation that can lead to exploitable security vulnerabilities. So a CWE description provides details and examples on poor software designs that can lead a software system to be subject to a CVE. Some CVE entries are linked to one or more CWE entries.

Patterns of use of vulnerabilities are captured in the Common Attack Pattern Enumeration and Classification (CAPECTM) [76] database, where attack prerequisites, outcomes, indicators, execution flow, severity, solutions and mitigations, attacker skills or knowledge required, and attack variations are captured. Used together, CWE and CAPEC provide a complete viewpoint on where and how software is likely to be attacked.

The amount of information provided by this network of repositories covers several important areas for understanding risks in software design: (i) typical attack patterns and vulnerabilities they exploit, (2) existing preconditions on software design, (ii) the cost of exploitation and access of a vulnerability, and (iii) the typical impacts of the vulnerabilities. An open challenge is how to create a similar infrastructure for *cyber-physical* systems. The principal limitation we should overcome is lack of disclosure of information on commercial HW/SW components.

3.3.1 Mitigation/Exploitation Cost of Vulnerability Exploits

We consider two different costs: the cost for the attacker to exploit an attack due to a vulnerability of the system (*exploitation cost*), and the cost to mitigate/fix the vulnerability and prevent the attacks associated to it (*mitigation cost*). Information on the exploitation cost could be derived from the vulnerability databases described in the previous section. The estimate of the mitigation cost is depending from a number of factors that are out of scope for the current discussion, including the estimation of the value of the asset to be protected.

We now discuss metrics provided by the CVSS system and can be used to define an estimate of the exploitation cost of an attack:

- *Severity base score*, estimates the severity of a CVE. An attacker is likely to put more effort in exploiting high-severity vulnerabilities. The severity of the CVSS (version 3.0) is divided into four categories: low (0.0–3.9), medium (4.0–6.9), high (7.0–8.9), critical (9.0–10.0). This value is calculated as a function of other metrics: exploitability, scope, and impact.
- *Exploitability score*, is the most important factor in the equation to calculate the exploitation cost. It estimates how easy it is for an attacker to exploit a vulnerability through an attack. The CVSS exploitability score relies on the following metrics.

Attack vector, the context by which the exploitation is possible.

Attack complexity, considers the conditions, beyond the attacker control, that must exist to exploit the vulnerability.

Privileges required, determines the level of privileges an attacker must have to exploit the vulnerability.

User interaction, determines if the vulnerability requires user involvement or collaboration to be exploited.

- *Scope (or authorization scope)*. A Boolean value used to estimates if a vulnerability can impact resources beyond its means of privileges.
- *Impact score*. Estimates the impact on the core confidentiality, integrity and availability security properties.

Considering cyber-physical systems, there is a number of areas where research on the definition of adequate metrics. For example, we should include the physical distribution of the system, the physical accessibility, the effort to influence the state of a plant and disguise supervisory controls. On the other side, the potential impacts are enormous and therefore be motivating for higher investment.

3.3.2 A High-Level Representation of Vulnerabilities

A first approximation for representing vulnerabilities for the purpose of high-level Security Risk Assessment is to describe (1) the preconditions that allow the exploitation of the vulnerability by the attacker, and (2) how it impacts the

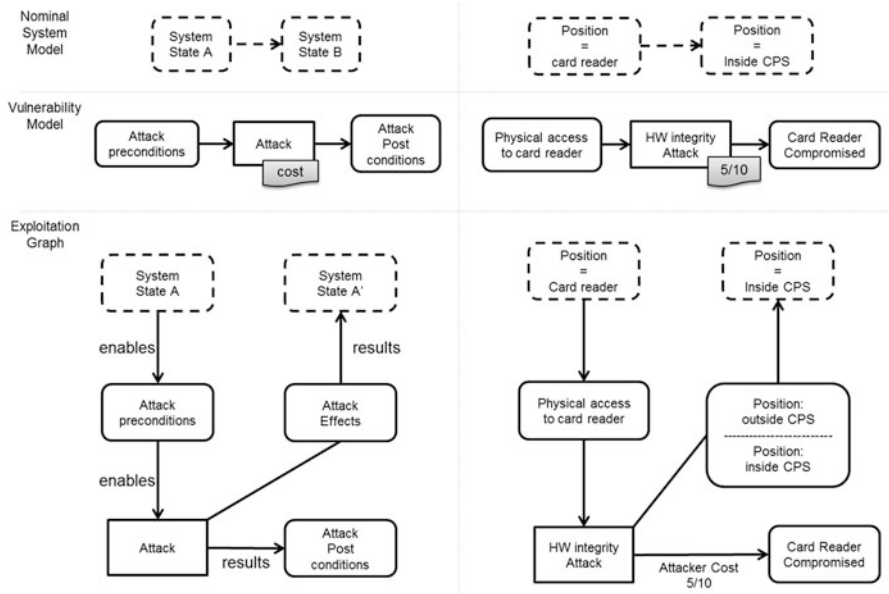


Fig. 3 (left) Exploitation graph and (right) its implementation in a toy example

system. An exploit can be used whenever the preconditions are in place in the current status of the system. However, the attacker does not use all the exploits whenever he can, but he considers the cost of using an exploit and tries to select the cheapest combination of exploits that reaches the attacker’s goals. The impact of the exploitation of vulnerabilities in CPS is a deviation of the system behavior from the nominal one, leading to damages to the system assets.

If we assume the nominal system behavior is represented as a transition system an exploit can be modeled as an enriched transition from a nominal state of the system model to a new state, enabled by the exploitation of a vulnerability. As depicted in Fig. 3 (left), a vulnerability is enabled when specific attack preconditions are satisfied in a system state. The vulnerability, once exploited, results in a set of post-conditions, whose effects entail a variation of the system state. Whenever a vulnerability exploit is executed, the attack cost is incremented and a new system state is reached.

As an example, in Fig. 3 (right), we assume that an access control panel requires demonstrating valid credentials through use of a card to access to a restricted maintenance room of a CPS. The vulnerability is modeled so that whenever the attacker has physical access to the authentication panel (precondition), he can exploit an integrity vulnerability (attack), e.g. by physical tampering of the card reader, and gain access to the CPS. The post-conditions are expressed in terms of the effects of the exploitation of the attack, i.e., the changing of the physical position

of the attacker. This is a deviation from the system nominal behavior that expects the access to be granted only to persons that own a badge.

4 A Vision for CPS Security Risk Assessment

We now provide a summary of the challenges and opportunities discussed so far in the form of a high-level description of how we envision SecRA to be performed for CPS. The purpose is to stimulate further discussions and to serve as a starting point for the definition of a research roadmap on this novel and challenging area. The summary we propose leverages practices that are already adopted in safety-relevant and high-assurance systems (e.g. model-based design flows [19], structured approaches to security and safety from the automotive and aerospace domains [8, 9], and standards for Cyber-security Risk Assessment [5]) but requires to address multiple challenges before reaching the required level of automation and formalism.

1. *Secure system architecture and behavior design* is structured into progressive refinement steps, supported by models, and organized into viewpoints:
 - 1.2 *Refinement steps*: (ref. to Fig. 1) the design starts from the definition of a *concept*, where there is no notion of security and the principal system functions are identified and allocated to a high-level physical architecture, then is refined into a *functional security architecture*, where (given a decomposition of system functions into component functions and a refinement of the physical architecture by defining interfaces and data-flows) the security measures and controls are identified on the basis of a preliminary SecRA, then it is finalized into a *technical security architecture*, where (given a complete definition of system functions, physical components interfaces and data-flows) the security measures and controls are defined in details on the basis of the results of SecRA.
 - 1.3 *Models*: should cover both the architecture and the behavior
 - 1.3.1 *Architecture*: shall be used to understand the attack surface, the attack paths, the location of security controls and measures, the layers of security to design defense in depth.
 - 1.3.2 *Behavior*: shall be modeled to support automated extraction of threat scenarios, to support simulation of identified threat scenarios, and also to allow better run-time security measures and controls.
 - 1.4 *Viewpoints*: shall be used to decompose the system modeling activity into teams (function, safety, security, validation, environment, etc.), to manage complexity of the overall design, and to leverage work shared across team.
2. *Adoption of Formal Behavioral Models*: the adoption of formal models can be challenging, require specific expertise, and cause an extra effort in the process.

However, it opens up the opportunity of performing automated and exhaustive analyses, where the provided assurance is higher than with other methods. It requires the use of specific modeling languages, supported by formal analysis tools, that are typically very effective to model digital systems and have currently partial support for complex, physics-based models. For all these reasons, we envision the use of formal and automated analyses for SecRA either at the level of the functional security architecture, where more abstract models can be used, or for the Risk Assessment of specific high-criticality components.

3. *Compositionality and Reuse*: the architecture and behavioral models should be developed in a compositional approach (ref. to Fig. 2), to allow replacement and reuse of different models to construct multiple versions of the *Nominal System Model*, defining the system behavior under normal conditions. Compositionality applies also to the *Attacker Model* since, depending on the asset we should protect and on the environment the system shall operate in, we envision reusing the nominal system behavior model, and composing it with different attacker profiles. Similar considerations apply to the *Cost and Impact Model* and, clearly to the *Environment Model*. Finally, we envision the need of defining and maintain a *Library of Vulnerability Models* that should be applicable by composition to the single components and have the role of changing the nominal component model to a *Component Under Attack Model*. This component-driven extension approach shall support the derivation of a *System Under Attack Model* without additional effort.
4. *Security Goals and Risk Assessment*: we take an asset-driven approach to SecRA, which (in our experience) allows us to better define the security goals in terms of minimizing the loss of value of the assets. Security Goals should be formulated in terms of description of events that have an impact on the assets (*Incidents*) and should be avoided. Asset values can be used to provide a quantitative ranking of the Incidents. Therefore, formal analysis can be used to evaluate multiple scenarios that can lead to an Incident by leveraging known vulnerabilities of the system. A *cost/benefit analysis* can provide a ranking of the different attacks, in terms of the cost they require from the attacker and the impact they can cause on Assets. From this analysis, we expect to extract the most critical risks and perform a root-cause analysis to determine what vulnerabilities are more influential and shall be mitigated.

As discussed in Sect. 2.2, there are several existing languages and toolchains that can support the System Architecture and Behavior Design step described previously. A notable example is SysML, which embeds the notion of viewpoint natively, supports compositional definition of behaviors by Sequence Diagrams or Statecharts [77], and is implemented in several toolchains, that typically allow exporting models to perform analyses leveraging external tools [78].

Concerning the modeling of component vulnerabilities, as described in the Compositionality and Reuse step, there are several challenges and opportunities, described in Sect. 3.3. Our current vision on how to extend the system nominal behavior has been described in Sect. 3.3.2. The effect of a vulnerability (which

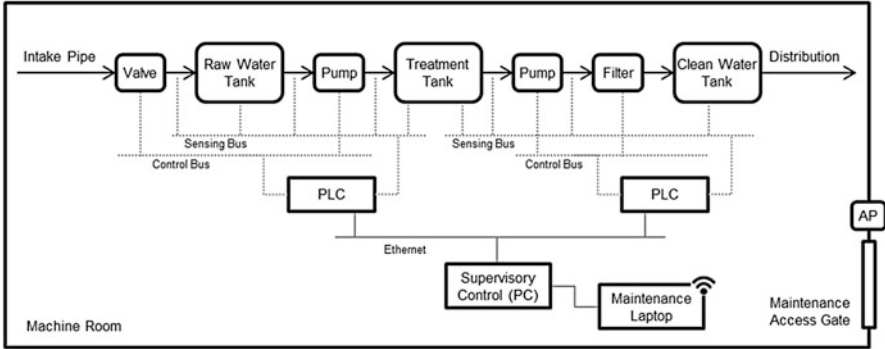


Fig. 4 WTP physical architecture

can be extracted from public repositories such as NVD) is expressed in terms of behavioral changes of the nominal system model or as changes to the knowledge of the attacker. The latter case can lead to a reduction of the cost associated to the exploitation of vulnerabilities, thus influencing the Cost/Impact Model.

Finally, there is an interesting aspect to be discussed on the Security Goals and Risk Assessment step. Security Goals should capture conditions that avoid Incidents to Assets. Our current approach on this aspect is to formally define Security Goals as invariants with an associated numeric value and a related Asset. Violation of an invariant represents a state of the system where an Incident occurred and damage has been caused to an Asset with a specific value loss. This allows ranking attacks in terms of the value loss they produce. Let us now consider an example where some of the ideas discussed so far can be seen in practice.

Example We consider a Water Treatment Plant (WTP) as in Fig. 4, completely disconnected from the Internet, during operation, located in a Machine Room physically accessible through a Maintenance Access Gate through an Authentication Panel (AP). The WTP is periodically maintained by a technician, who connects his laptop to the Supervisory Control PC to collect performance and diagnosis data, which are sent back to the WTP production company, both for diagnosis purposes and for performance analysis.

Similarly to [79], inside the WTP there is an initial tank where the raw water is stored. The tank is connected via pipes to several intermediate steps where the water is purified by means of chemicals which are then filtered by filtering procedures (e.g., Ultraviolet filters for dechlorination). The *principal asset* we consider in this example is the physical integrity of the raw water tank. Two important *indirect assets* are (a) safety of technicians (possibly in the premises of the plant) and (b) building integrity, which may be compromised by the effect of burst and water dispersion. For the sake of simplicity, we suppose there are only the following threats (summarized in Table 1):

1. *Pressure safety control*, the WTP system does not have a safety control against the increase of pressure in the tank above a critical threshold. Therefore, if attacker is able to change the pressure of the raw water tank then he can lead the tank, eventually, to burst. Considering the accessibility of the WTP, its architecture, and the fact that controllers are not connected to the Internet, the only way for the attacker to increase the pressure is by acquiring physical access to the WTP and manipulate the valves and pumps of the system. The *cost for the attacker* is high since (we supposed) access to the plant requires valid credentials (which the attacker does not know). The *mitigation cost* is high too (8/10) because it requires the installation of an intrusion detection system (plus additional costs to manage the system) and/or the introduction of authentication/encryption schemes in the communication between components. We note that the assumptions on both the vulnerability and the costs are fair since similar situations have been reported on real-world water treatment testbed and plants (see, e.g., [80]).
2. *Ethernet local network*, since the local network is isolated from the Internet, there are no security protocols implemented to guarantee the basic security properties (i.e., confidentiality, integrity, availability). An attacker can easily modify the content of the packets exchanged between PLC. We assume (as in [81]), that the logic of the control of valves and pumps is implemented in the Supervisory Control PC while PLC converts analog to digital messages. Therefore, if the attacker has access to the Ethernet network (similarly to [80]), he can modify the payload of the network packets affecting the control logic of valves and pumps. Considering the accessibility of the WTP, and the isolation from the Internet, the *cost for the attacker* is high. The *mitigation costs* are high too, since the introduction of security mechanisms that guarantee the basic security properties requires the re-engineering of the local network. Furthermore, those security mechanisms need to take into account the timing constraints such as the safety response time of the WTP that may result in ad-hoc solutions or delicate fine-tuning of security protocols.
3. *Authentication*, the storage room hosting the WTP can be accessed through authentication on a panel. The panel is subject to (a) physical tampering, (b) attacks through accessible port used for firmware updates, and (c) remote attacks through the access control management server. An attacker that is able to exploit any of these vulnerabilities can gain access to the room. We assume that for an attacker can be relatively easy to bypass authentication procedures since there is a wide number of vulnerabilities and social engineering techniques that have been effectively exploited in the past decades. For the same reason, properly mitigate this risk is not trivial but a number of techniques can be used to mitigate this risk.
4. *USB*, disabling USB ports in a computer connected to a plant increases the security of the overall system with a relative cheap cost of finding other methods to transfer data into the system. On the other hand, if not disabled, many examples (e.g., [17]) showed that USB can be easily exploited by attackers to get access to the system and/or to introduce malwares.

Table 1 Exploitation and mitigation cost in isolation

Threats	Attacker cost	Mitigation cost
Pressure changing	10/10	8/10
Ethernet integrity	9/10	9/10
Authentication bypass	4/10	6/10
USB (SCADA system)	8/10	2/10

An example multi-stage attack that can be identified by the application of a model-based SecRA mixes cyber and physical vulnerabilities with ascending attacker cost.

1. The attacker access to the WTP exploiting social engineering attacks (e.g., piggybacking or tailgating) or integrity vulnerability exploits of the AP.
2. Once the attacker has access to the CPS, he can exploit the USB vulnerability to get access to the network of the plant.
3. Since there are no security mechanisms on the Ethernet network, the attacker can easily alter the payloads of the messages.
4. The attacker alters the nominal behavior of the system (e.g., opening the intake valve and closing the pump after the raw water tank) and burst the water tank.

Such attack shows that the exploitation cost in isolation of Table 1 should change after the discovery of attack chains. In fact, Table 1, the attacker exploitation cost of the USB, and Ethernet threats is high because based on the assumption that access to the CPS is forbidden. However, the attack shows that the attacker cost of exploiting USB, or Ethernet threats is linked to the authentication bypass cost.

Summarizing, instead of estimating the risk as the vulnerability exploitation cost in isolation, model-based analysis techniques can be applied leveraging system architecture and nominal behavior models extended with vulnerabilities. In this way, one could discover complex attack chains and estimate the risks more precisely, based on the interconnection between different vulnerabilities. In our example, the authentication bypass vulnerability has the highest risk because, if not mitigated, can be used to lower the exploitation cost of all the other vulnerabilities.

5 Conclusion

We discussed challenges and opportunities to develop a model-based SecRA for CPS, adopting an asset-driven viewpoint to evaluate risks at system-level, and we identified gaps. We considered modeling languages, techniques, and tools for the SecRA, discussing limitations in their direct application to a CPS. We defined a roadmap of research opportunities for (i) CPS (security) modeling, (ii) vulnerabilities and cost models, (iii) integration with public vulnerability repos, (iii) attacker models and profiles, and (iv) automated discovery of cyber-physical attack chains.

References

1. Sampigethaya K, Poovendran R (2013) Aviation cyber-physical systems: foundations for future aircraft and air transport. *Proc IEEE* 101(8):1834–1855
2. Moir I, Seabridge A, Jukes M (2013) *Civil avionic systems*. Wiley, Hoboken
3. Shavit M, Gryc A, Miucic R (2007) Firmware update over the air (FOTA) for automotive industry. In: *Asia Pacific automotive engineering conference*.
4. Howard M, Lipner S (2006) *The security development lifecycle*, vol 8. Microsoft Press, Redmond
5. ISA/IEC 62443 *Security for industrial automation and control systems*
6. Disterer G (2013) ISO/IEC 27000, 27001 and 27002 for information security management. *J Inf Secur* 4(2):92–100
7. Joint Task Force Transformation Initiative (2003) SP 800–53 Rev. 4, NIST
8. RTCA Inc (2014) DO-356. RTCA
9. SAE (2016) J3061 – Surface vehicle recommended practice. SAE International technical report
10. ISO/IEC 15408. *Information technology – security requirements – evaluation criteria for IT security*
11. The CORAS EU Project FP5 IST-2000-25031, FP5-IST
12. Blanchard BS, Fabrycky WJ, Fabrycky WJ (1990) *Systems engineering and analysis*. Prentice Hall, Englewood Cliffs
13. Rumbaugh J, Jacobson I, Booch G (2004) *Unified modeling language reference manual*, 2nd edn. Pearson Higher Education, Peking
14. Schneier B (1999) Attack trees. *Softw Tools Prof Progr* 24(12):21–29
15. Shmeli-Sendi A, Aghababaei-Barzegar R, Cheriet M (2016) Taxonomy of information security risk assessment (ISRA). *J Comput Secur* 57(C):14–30
16. Shi J, Wan J, Yan H, Suo H (2011) A survey of cyber-physical systems. In: *International conference on Wireless Communications and Signal Processing (WCSP)*
17. Weinberger S (2011) Computer security: is this the start of cyberwarfare? *Nat News* 474(7350):142–145
18. Miller B, Rowe D (2012) A survey SCADA of and critical infrastructure incidents. In: *Proceedings of the conference on research in information technology*
19. Edwards S, Lavagno L, Lee E, Sangiovanni-Vincentelli A (1997) Design of embedded systems: formal models, validation, and synthesis. *Proc IEEE* 85(3):366–390
20. West A (2009) *Nasa study on flight software complexity*. NASA
21. OWASP, The Open Web Application Security Project (OWASP) [Online]. Available: www.owasp.org. Accessed Sept 2017
22. MITRE, Common Weakness Enumeration (CWE) [Online]. Available: cwe.mitre.org. Accessed Sept 2017
23. Hutchins EM, Cloppert MJ, Amin RM (2011) Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Lead Issues Inf Warf Secur Res* 1(1):80
24. Tankar C (2011) Advanced persistent threats and how to monitor and deter them. *Netw Secur* 2011(8):16–19
25. Ict, Sintef, The CORAS method [Online]. Available: <http://coras.sourceforge.net/>
26. ThreatModeler [Online]. Available: threatmodeler.com. Accessed Sept 2017
27. Microsoft Corporation, STRIDE – threat modeling [Online]. Available: <https://msdn.microsoft.com/en-us/library/ff648644.aspx>
28. RTCA (2011) DO-333 – formal methods supplement to DO-178C and DO-278A. RTCA
29. Blanchet B (2012) Security protocol verification: symbolic and computational models. In: *International conference on Principles of Security and Trust (POST)*
30. Refsdal A, Solhaug B, Stolen K (2015) *Cyber risk management*. In: *Cyber risk management*. Springer, Cham, pp 33–47

31. International Organization for Standardization (2009) ISO 31000 – risk management – principles and guidelines
32. Cherdantseva Y, Burnap P, Blyth A, Eden P, Jones K, Soulsby H, Stoddart K (2016) A review of cyber security risk assessment methods for SCADA systems. *Comput Secur* 56(C):1–27
33. NIST, Cybersecurity framework [Online]. Available: <https://www.nist.gov/cyberframework>
34. Shostack A (2014) *Threat modeling: designing for security*. Wiley, Indianapolis
35. SESAR [Online]. Available: www.sesarju.eu
36. Lund MS, Solhaug B, Stølen K (2011) *The CORAS approach*. Springer, Berlin/Heidelberg
37. NIST, National Vulnerability Database (NVD) [Online]. Available: nvd.nist.gov. Accessed Sept 2017
38. OMG, UML success stories [Online]. Available: http://www.uml.org/uml_success_stories/index.htm. Accessed Sept 2017
39. Houmb SH, Den Braber F, Lund MS, Stølen K (2002) Towards a UML profile for model-based risk assessment. In: *Workshop on critical systems development with UML*
40. Lund MS, Hogganvik I, Seehusen F, Stølen K (2003) UML profile for security assessment. Technical report STF A
41. Lodderstedt T, Basin D, Doser J (2002) SecureUML: a UML-based modeling language for model-driven security. In: *Proceedings of the international conference on the unified modeling language*
42. Jürjens J (2002) UMLsec: extending UML for secure systems development. In: *Proceedings of the international conference on the unified modeling language*
43. McDermott J, Fox C (1999) Using abuse case models for security requirements analysis. In: *Proceedings of Computer Security Applications Conference (ACSAC)*
44. Sindre G, Opdahl AL (2005) Eliciting security requirements with misuse cases. *Requir Eng* 10(1):34–44
45. Weillkiens T (2007) *Systems engineering with SysML/UML: modeling, analysis, design*. The OMG Press, Amsterdam/Boston
46. Roudier Y, Apvrille L (2015) SysML-sec: a model driven approach for designing safe and secure systems. In: *Model-Driven Engineering and Software Development conference (MODELSWARD)*
47. Lugou F, Li LW, Apvrille L, Ameur-Boulifa R (2016) Sysml models and model transformation for security. In: *Model-Driven Engineering and Software Development conference (Model-sward)*
48. E-safety Vehicle Intrusion Protected Applications (EVITA) EU FP7 Programme, 2007–2013
49. AADL [Online]. Available: <http://www.aadl.info/>. Accessed Mar 2018
50. Ellison R, Householder A, Hudak J, Kazman R, Woody C Extending AADL for security design assurance of cyber-physical systems. CMU/SEI-2015-TR-014
51. Rocchetto M, Tippenhauer NO (2017) Towards formal security analysis of industrial control systems. In: *Asia conference on Computer and Communications Security (AsiaCCS)*
52. Ahmed CM, Murgia C, Ruths J (2017) Model-based attack detection scheme for smart water distribution networks. In: *Asia conference on Computer And Communication Security (AsiaCCS)*
53. Rocchetto M, Tippenhauer NO (2016) On attacker models and profiles for cyber-physical systems. In: *European symposium on Research in Computer Science (ESORICS)*
54. Lanotte R, Merro M, Muradore R, Viganò L (2017) A formal approach to cyber-physical attacks. In: *Computer Security Foundation symposium (CSF)*
55. Herley C (2016) Unfalsifiability of security claims. *Natl Acad Sci* 113(23):6415–6420
56. Blanchet B (2016) Modeling and verifying security protocols with the applied pi calculus and ProVerif. *Found Trends Priv Secur* 1(1–2):1–135
57. Garland D (2003) Formal modeling and analysis of software architecture: components, connectors, and events. In: *Proceedings of formal methods for software architectures*
58. Schmidt M, Lipson H (2009) Distilling free-form natural laws from experimental data. *Science* 324(5923):81–85

59. Schupp S, Abraham E, Chen X, Makhlouf IB, Frehse G, Sankaranarayanan S, Kowalewski S (2015) Current challenges in the verification of hybrid systems. In: *CyPhy 2015*, LNCS 9361, pp 8–24
60. Platzer A (2010) *Logical analysis of hybrid systems*. Springer, Berlin/Heidelberg
61. de Moura L, Bjørner N (2008) Z3: an efficient SMT solver. In: *Tools and Algorithms for the Construction and Analysis of Systems conference (TACAS)*
62. Jovanović D, de Moura L (2012) Solving non-linear arithmetic. In: *International Joint Conference of Automated Reasoning (IJCAR)*
63. Dutertre B (2014) Yices 2.2. In: *Computer Aided Verification (CAV)*
64. Cimatti A, Griggio A, Schaafsma BJ, Sebastiani R (2013) The MathSAT5 SMT solver. In: *Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*
65. Barrett C, Conway CL, Morgan D, Hadarean L, Jovanović D, King T, Reynolds A, Tinelli C (2011) Cvc4. In: *International conference on Computer Aided Verification (CAV)*
66. Cimatti A, Griggio A, Irfan A, Roveri M, Sebastiani R (2017) Invariant checking of NRA transition systems via incremental reduction to LRA with EUF. In: *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*
67. Dolev D, Yao A (1983) On the security of public key protocols. *IEEE Trans Inf Theory* 29(2):198–208
68. Escobar S, Meadows C, Meseguer J (2006) A rewriting-based inference system for the nrl protocol analyzer and its meta-logical properties. *Theory Comput Sci* 367(1–2):162–202
69. Basin D, Capkun S, Schaller P, Schmidt, B (2009) Let’s get physical: models and methods for real-world security protocols. In: *International conference on Theorem Proving in Higher order Logics (TPHOL)*
70. Barik MS, Segupta A, Mazumdar C (2016) Attack graph generation and analysis technique. *Def Sci J* 66(6):559–567
71. Wang JA, Guo M (2009) Ovm: an ontology for vulnerability management. In: *Workshop on Cyber Security and Information Intelligence Research (CSIIRW)*
72. Felderer M, Zech P, Breu R, Büchler M, Pretschner A (2016) Model-based security testing: a taxonomy and systematic classification. *Softw Test Verif Reliab* 26(2):119–148
73. Mell P, Scarfone K, Romanosky S (2006) Common vulnerability scoring system. *IEEE Secur Priv* 4(6):85–89
74. Mell P, Grance T (2002) Use of the common vulnerabilities and exposures (cve) vulnerability naming scheme. National Institute of Standards and Technology, Computer Security Division, Gaithersburg MD
75. MITRE, Open Vulnerability and Assessment Language (OVAL) [Online]. Available: <https://oval.cisecurity.org/>. Accessed Sept 2017
76. MITRE, Common Attack Pattern and Enumeration and Classification (CAPEC) [Online]. Available: <http://capec.mitre.org/>. Accessed Sept 2017
77. Glinz M (1995) An integrated formal model of scenarios based on statecharts. In: *Software Engineering (ESEC)*
78. Arnold A, Baleani M, Ferrari A, Marazza M, Senni V, Legay A, Quilbeuf J, Etzien C (2016) An application of SMC to continuous validation of heterogeneous systems. In: *SimuTools, ICST, Brussels, Belgium*
79. Mathur AP, Tuppenhauer NO (2016) SWaT: a water treatment testbed for research and training on ICS security. In: *Proceedings of the cyber-physical systems for smart water networks (CySWater) workshop*
80. Urbina D, Giraldo J, Tuppenhauer NO, Cardenas A (2016) Attacking fieldbus communications in ICS: applications to the SWaT Testbed. In: *Proceedings of Singapore Cyber security conference (SG-CRC)*
81. Rocchetto M, Tuppenhauer NO (2016) CPDY: extending the Dolev-Yao attacker with. In: *International Conference on Formal Engineering Methods (ICFEM)*

A Comprehensive Framework for the Security Risk Management of Cyber-Physical Systems



Hassan Mokalled, Concetta Pragliola, Daniele Debertol, Ermete Meda, and Rodolfo Zunino

Abstract Cyber Physical Systems are facing huge and diverse set of security risks, especially cyber-attacks that can cause disruption to physical services or create a national disaster. Information and communication technology (ICT) has made a remarkable impact on the society. As a Cyber Physical System (CPS) relies basically on information and communication technology, this puts the system's assets under certain risks especially cyber ones, and hence they must be kept under control by means of security countermeasures that generate confidence in the use of these assets. And so there is a critical need to give a great attention on the cybersecurity of these systems, which consequently leads to the safety of the physical world. This goal is achieved by adopting a solution that applies processes, plans and actions to prevent or reduce the effects of threats. Traditional IT risk assessment methods can do the job, however, and because of the characteristics of a CPS, it is more efficient to adopt a solution that is wider than a method, and addresses the type, functionalities and complexity of a CPS. This chapter proposes a framework that breaks the restriction to a traditional risk assessment method and encompasses wider set of procedures to achieve a high level strategy that could be adopted in the risk management process, in particular the cybersecurity of cyber-physical systems.

Keywords Cyber-physical system · Risk management · Cybersecurity · Ansaldo STS

H. Mokalled (✉)

Ansaldo STS, Cyber Security Assurance & Control Department, Genoa, Italy

University of Genoa, DITEN, Genoa, Italy

Lebanese University, EDST-MECRL Lab, Beirut, Lebanon

C. Pragliola · D. Debertol · E. Meda

Ansaldo STS, Cyber Security Assurance & Control Department, Genoa, Italy

R. Zunino

University of Genoa, DITEN, Genoa, Italy

e-mail: rodolfo.zunino@unige.it

© Springer International Publishing AG, part of Springer Nature 2019

F. Flammioni (ed.), *Resilience of Cyber-Physical Systems*,

Advanced Sciences and Technologies for Security Applications,

https://doi.org/10.1007/978-3-319-95597-1_3

1 Introduction

A cyber-physical system refers to the system that combines both cyber and physical resources, where there is a strong relation and coordination between these resources. Such systems are controlled or monitored by computer-based algorithms, tightly integrated with the internet and its users. CPS is basically a control system with distributed networked, adapted and predictable, real-time, intelligent characteristics, where human-computer interaction may exist. It is widely used in critical national infrastructure, such as electric power, petroleum and chemical and so on [1]. Moreover, many urban transportation and railway systems around the world have deployed some form of communications-based automatic train control (e.g., [2]). And in those systems, multiple cyber components exist, including wireless communication. The potential implications of this evolution could be multi-faceted and profound, especially when it comes to the issue of security. If such systems were subject to a physical or cyber threat, the consequences will be unimaginable. These systems are susceptible to different types of risks related to information systems vulnerabilities. No one doubts about the hazardous consequences that would occur in case a malicious software succeeds in controlling the system, i.e. any fail in systems controlling drive-less metros will lead to huge loss. Security breaches in the cyber domain, such as falsified information or malicious control logic, can have a complicated impact on the physical domain [3]. “The cyber breach will lead to complicated physical consequences”. Cybersecurity breaches can range from no or limited impact to Distributed Denial of Services (DDoS), stealing of data, or even taking over control of systems and harm the physical world [4]. In energy industry, the computer system of Iran Bushehr nuclear power plant was invaded by “Stuxnet” in 2010, leading a serious chaos in the automated operation of the nuclear facilities and a serious setback of Iran’s nuclear program. In transport service, in the network for managing and monitoring the operation of the Shinkansen, due to an exception in the management system of control schedule, signaling and line switching point in 2011, Japan’s 5 Shinkansen operation management system encountered failure, 15 trains were in outage, 124 trains were delayed and 8.12 million people’s travel were affected. In water Industry, in 2011, Illinois water system was hacked and a malfunction occurred in the water pump SCADA, which leading to the pump’s damage and scrap. In this way, we can conclude that CPS security is so important that risk incidents in the system may affect national security and stability. Taking all these security incidents seriously, we conclude that any attack in the cyber layer of the cyber physical system could lead to hazardous situations and even to loss in lives [1].

There are several approaches for the problem of risk assessment and treatment: informal handbooks, methodical approaches or supporting tools, where all provide a guide for risk assessment and treatment. However, methods might differ in some steps, or in the way of identifying and valuating the assets or threats. Some are basically used in cyber security of information systems, and others can be used in physical security. Many of the proposed solutions try to measure or estimate

the probability and the severity of the risks after identifying the assets and threats using traditional IT risk assessment methods, some of these solutions do not address the characteristics and the complexity of CPS, which needs a broad range of management. The great challenge of these approaches is the complexity of the problem they have to face; in the sense that there are many elements to be considered and, if it is not done rigorously, the conclusions will be unreliable.

Ansaldo STS is a leading Company operating in the sector of high technology for Railway and Urban Transport. The Company has the experience and resources to supply innovative transport and signaling systems for freight yards, regional and freight lines, underground and tramway lines, and standard and High-Speed railway lines. With an international geographical organization, the Company operates worldwide as lead contractor, system integrator and supplier “turnkey” of the most important projects of mass transportation in metro and urban railways. Ansaldo STS has a great experience in the design, implementation and management of systems and services for signaling and supervision of railway and urban traffic [5].

Ansaldo STS believes that there is a critical need to adopt a comprehensive strategy for the problem of applying risk management study to a cyber-physical system. As the complexity of the CPS is greater and such systems need more procedures to be performed, a framework was developed that aims to reach a common high level solution, it is different and broader than a traditional IT risk management methods whose goal is mainly focused on identifying and measuring the severity of the risks and try to reduce it to an acceptable extent. In fact, it encompasses seven steps and inspired by the PDCA cycle, and centered upon the cyber side and its assets; however, this doesn't mean that the physical assets are out of the frame, as the physical assets of a CPS are mostly controlled by others in the cyber side. This framework is characterized by a set of procedures that starts by modeling the system's assets and functionalities, selection of potential threats to the CPS, conducting risk assessment and treatment through a methodical way, safeguard implementation, vulnerability assessment, ensuring the compliance with global and local applicable laws, and finally applying maintenance and improvement activities. This chapter is divided as follows: Sect. 2 presents a set of aspects that the approach mentions, Sect. 3 describes the proposed framework. Section 4 is the case study that shows how Ansaldo STS Company applies this framework, and finally Sect. 5 concludes the work.

2 Aspects and Requirements

2.1 Cyber Physical System Security

CPS security has some distinct characteristics as a CPS is different from traditional IT system. In traditional IT systems the first important aspect of information security is confidentiality. Confidentiality means the protection of data, providing access for

those who are allowed to see it while disallowing others from learning anything about its content. However for CPS, the availability comes first, then integrity and confidentiality.

CPS has more attack points and fault points than IT system. Any safeguard measures shall not interrupt the response to the physical system or delay the response. In traditional IT system access control can be deployed without affecting the services of IT system. In CPS all these measures should be discussed and tested to great details. The data flow shall not be hindered or interfered. CPS is a system of systems, the tight coupling between the physical system and cyber system has led to potential cascade effect of the whole system. Malfunction whether in cyber part or in the physical part will spread to other part of system [1].

2.2 Threats and Vulnerabilities

The two main kinds of threats that affect any organization are internal and external threats. Internal threats occur from within the organizations. This is probably one of the most dangerous situations because for instance co-workers may know how to access the systems and are aware of how the systems are set up. And external threats are attacks done by externals and hackers [6].

- (i) **Internal Threats:** Statistics [7, 8] show that a large amount of security and privacy breaches are due to insiders. Protection from insider threats is challenging because insiders may have access to many sensitive and high-privileged resources. Similar style of exploitation is reported in [9, 10].
- (ii) **External threats:** External threats are those done by individuals from outside a company or organization, who seek to break defenses and exploit vulnerabilities. Spying or eavesdropping, DoS, Spoofing, Phishing, viruses, etc. . . . , are all examples of external threats or cyber-attacks.

On the other hand **vulnerability** is defined as a weakness in the system assets or safeguards that facilitates the success of a potential threat and could cause damage; they could exist in system, software, network, etc . . .

2.3 Security Requirements

The cyber security of a CPS calls for the use of a wide set of security controls to protect the whole system against compromises of their Confidentiality, Integrity and Availability (CIA). The cybersecurity of CPS must address these main security requirements:

- (i) **Integrity:** It means that only the authorized users can change in the assets, it is satisfied if the assets are not changed by an unauthorized party.

- (ii) **Confidentiality**: This means that the assets must not be exposed to unauthorized individuals, and the access must be restricted to those authorized. This is satisfied if the assets are not read or accessed by an unauthorized party.
- (iii) **Availability**: This is satisfied if the assets or services are available and without delay.

If the system was exposed to malicious activities, physical components would also be affected and even damaged as a consequence. It can be said that in a CPS, the availability comes first, then the integrity and confidentiality.

2.4 Dependencies and Accumulated Risk

As mentioned above, it is more efficient for a security strategy to start with functional modeling of assets with defining relations and dependencies, as it leads to more precise and coherent study. Dependencies affect all the calculations done to assess the risk. Since assets depend on each other, the occurrence of threats on assets causes a direct harm on them and an indirect harm on others that depend on them.

3 A Comprehensive Framework for the Risk Management–Cybersecurity in CPS

Commonly, when there is a need to assess risks, traditional methods are used to do the job. Traditional risk management methods involve the following step: risk identification, assessment and mitigation plan definition. However, a well-designed risk assessment of CPS will provide an overall view of CPS security status and support efficient allocations of safeguard resources. Though traditional IT system risk assessment is quite mature, a distinct risk assessment method for CPS is needed to cover the growing security issues due to the large differences between IT systems and CPSs [1]. This framework is inspired by the PDCA (PLAN-DO-CHECK-ACT) cycle. It adds a broader set of procedures for a traditional risk assessment method.

Companies must realize the necessity of managing data protection, they should better treat and manage the security strategy addressing the organizational and the technological aspects of the system [11], and also address the complexity and additional type of assets that a CPS encompass. In order to assure compliance with security and safety requirements, there is a need to define and adopt a holistic framework for risk assessment and treatment activities of CPSs, and so this section shows the proposed framework. Figure 1 shows how each step of the framework falls inside one of the phases of the PDCA cycle. It is divided into the following seven steps:

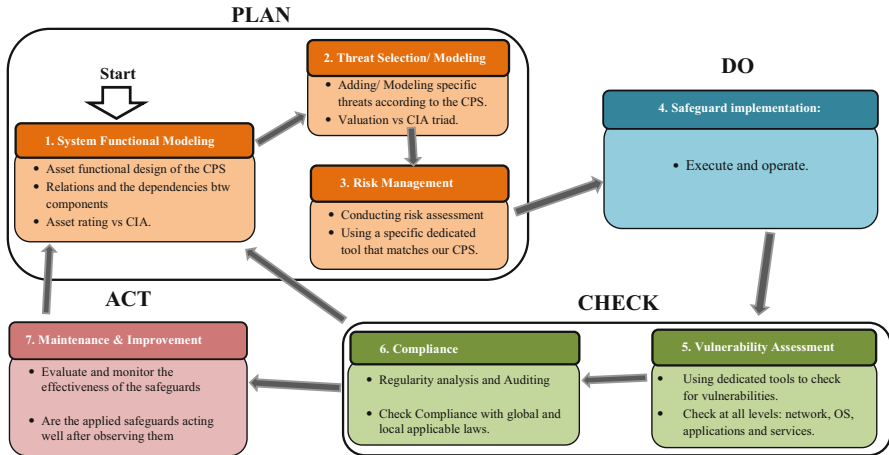


Fig. 1 The proposed framework inspired by the PDCA cycle

1. System Functional Modeling
2. Threat Selection and Modeling
3. Applying a Risk Management method (Assessment and Treatment plan)
4. Safeguard Implementation
5. Vulnerability Assessment
6. Compliance and Validation
7. Maintenance and Improvement

To ensure the continuous improvement, the framework is based on Deming PDCA Cycle where each phase, because of the complexity of a CPS, can be divided further in a few steps. The steps are applied in order: starting by the “PLAN” phase, first step is the “*System Functional Modeling*” which designs the model of the CPS showing the functionalities, dependencies, relations between the assets and defines also rules and Acceptable Risk Levels. Then the second step, “*Threat Modeling and Selection*” selects the potential “threats” that match the CPS’s assets: this can be done by referring to historical data such as reports, statistics, observations, logs, etc. Finally, always in the PLAN phase, the first two steps are the input to the “*Risk Management*” step, where an appropriate method is selected to assess the risk (Risk Assessment) and helps in selecting the appropriate measures for keeping the risks under control (Risk Treatment). After that “*Safeguard Implementation*” takes place, reflecting the “DO” phase of a PDCA, where the chosen decisions in the Plan phase are put into operation. Afterwards there is the CHECK phase, represented by the “*Vulnerability Assessment and Penetration Test*” process: it plays a key role in revealing the vulnerabilities yet present on the system and not protected by already installed safeguards. Because a CPS contains various set of HW/SW assets such as network appliances, servers, end-points, applications, web services, databases, etc., the Vulnerability Assessment and Penetration Test activity is applied basically

on three levels: Application, Network and Operation System Levels. Based on all previous findings and evidences, the CHECK phase is completed by a compliance control to ensure complying of the system to security best practices or international standards, e.g. ISO/IEC 27001/27002. Finally, the Deming Cycle is concluded by the ACT phase which contains “*Maintenance and Improvement*” activities to correct and improve the system.

3.1 System Functional Modeling (Asset Modeling)

Creating a functional model has a great impact in showing the structure and the components of the CPS, and in demonstrating the relations and the dependencies between the different assets, and hence to have a clear and precise simulation for the system in real life. It is the step where the whole framework depends on, in this stage it is meant to model the physical and cyber components and their interactions and operational characteristics. Asset Modeling can be considered as the most important step in this approach, it must be done first with the owners of the system. The scope of this part is to help the system’s owners or information sources in creating a system functional model and in the valuation of the system’s assets. For this task, two steps are followed:

- (i) Creating a functional model for the system which is a structured representation of the system’s components (assets) and functions (activities, processes, operations).
- (ii) Rating of the assets (based on CIA) using criticality levels and according to the consequences on CIA that would happen in case of their protection failure.

The two steps must be done by the owners or under the supervision of them. In this way, a typical representation or a general view for the system is carried out which aids in the risk management study.

3.2 Threat Selection and Modeling

Each CPS differs by the services and functionalities that it offers. Threats vary from one system to another, based on the available assets and their level of valuation. Different CPSs means different assets and though different types of threats. Threats can be grouped and associated to homogenous group of assets called asset classes. Threat selection is about understanding the most suitable threats that are expected to happen and matching them with the different asset classes of the cyber physical system. The appropriate threats-to-assets should be selected in this step to be fed into the “Risk Management study” step, and should be applicable to the assets presented in the previous step. Mainly cyber-security threats are covered; that is, threats applying to information and communication technology assets, but additional non-



Fig. 2 Common threats for the “Threat selection and Modeling” step in CPS

IT threats could also be included in order to cover threats to physical assets that are necessary for the operation of the CPS. This work can be done by referring to historical data, e.g.: reports, statistics, observations, logs, etc.

The ENISA Threat Landscape provides an overview of threats, together with current and emerging trends. It is based on publicly available data and provides an independent view on observed threats, threat agents and threat trends. Over 140 recent reports from security industry, networks of excellence, standardization bodies and other independent institutes have been analyzed [12], Fig. 2 shows a sample for some threats that threaten cyber physical systems. However risk analysts are the ones responsible for selecting and valuating the appropriate and expected threats that are likely to occur and match the system’s assets. First the general model is obtained by experts, reports, statistics, and then threats that match the context, type of the CPS and the given assets are kept and fed to the next step. Threat Modeling eases the risk analysis study in various ways, mainly it prepares a wealthy and substantial threats-to-assets convenient dataset that fits a case study. There are some dedicated tools that help in threat modeling, and Sect. 4.2 shows one of them which is used by Ansaldo STS Company.

3.3 Risk Management Plan

Risk management is divided into risk analysis and risk treatment, with risk analysis being the systematic process for estimating the risks to which the system's assets are exposed to [13]. Risk management is a part of planning, where treatment decisions are taken. These decisions are demonstrated and established in the implementation step.

1. **Risk analysis:** A risk is an indicator of what could happen to the assets if not properly protected. It is important to know what features are of interest in each asset and to what extent these features are in danger, that is, analyze the system [13]. There are several methods and ways for the problem of analyzing the risks: informal handbooks, methodical approaches or supporting tools, where all provide a guide for risk analysis. However, methods might differ in some steps, or in the way of identifying and valuating the assets or threats. Some are basically used in cyber security of information systems, and others can be used in physical security. Risk analysis study must be applied using an appropriate method and tool for the risk analysis step in the cybersecurity of CPSs. Applying a risk analysis study includes:
 - (i) Identifying and classifying assets by types, establishing dependencies between them and evaluating them according to security dimensions.
 - (ii) Identifying and valuating threats and their likelihood.
 - (iii) Identifying current safeguards and valuating them according to the level of effectiveness.
 - (iv) Evaluating the risk on the CPS system where valuations for assets, dependencies, and threats are all involved in the calculation.
2. **Treatment plan:** On the other hand, this sub-step must also carry out the risk treatment activities that should be applied. Risk treatment activities allow a security plan to be prepared which, when implemented and operated, meets the proposed objectives with the level of risk accepted by the Management. In the treatment plan, the right counter measures are selected with types, and then prioritized. Moreover defining their cost/complexity, effectiveness and efficiency metrics must be also addressed. The objective is to deploy the controls selected by type and in a prioritized and effective way. For example, same safeguard can contrast more threats at the same time and overlapping/redundant safeguards should be avoided. However, sometimes, when a series of safeguards are in place and the management process is mature to a certain extent, the system will still be exposed to a risk called "residual".

3.4 Safeguard Implementation: Operations

This step deals with the implementation of security plans and decisions taken in the treatment plan, it takes as input the activities defined and puts them into operation. It also deals more with the technical side, and defines the best technological solutions based on the countermeasures to be adopted and the approved budget in accordance with the defined strategy. Implementation of safeguards must ensure the availability and the capability of the organizational staff to manage the tasks scheduled to implement them, as well as other factors, such as the budget of the organization, relations with other bodies, legal, regulatory or contractual changes, etc. So applying security patches and ensuring the secure configuration of all appliances is maintained continuously, also assets are monitored and logs are analyzed to detect any improper actions. Even when the risks have been treated, residual risks will generally remain. Residual risk means that the current level of risk is accepted and is under a “carefully chosen” threshold, as trying to eliminate it could be extremely expensive.

3.5 Vulnerability Assessment

Vulnerability is a weakness in the assets that a malicious attacker could use to cause damage. Increasingly sophisticated tools help to penetrate existing network connections. After implementing the safeguards in the previous step, a vulnerability management process is needed to check if the assets of the cyber physical system are really still exploitable to threats. At the technical level, the focus is on cyber assets, this step is done by vulnerability exposure tools, with simulation of attack paths (similar to MITRE attack matrix). The end result can be patch management or better, in some complex environment, virtual patching (i.e. putting layer of defense that stop the attack before it reaches the endpoint, without the need to change configurations of the endpoint itself). Furthermore, log analysis could be useful in revealing vulnerabilities; but consider that doing manual log analysis requires a significant amount of expertise, knowledge, and is very time consuming. At the end, when detecting issues, it is required to return to the iteration cycles for proposals and solutions.

3.6 Compliance

Assessing the adherence of security configurations to the policies, requirements and regulations are set out in this stage. Compliance activities also involve regulatory analysis in order to ensure the compliance with global and local applicable laws based on the requirements, or even with respect to verification schemes to be achieved or maintained. And in case of non-compliance, it is required to return to the iteration cycles for proposals and solutions.

3.7 Maintenance and Improvements

Finally, the evaluation of the effectiveness and efficiency of the applied safeguards is measured to achieve the needed improvement and maintenance. It is recommended to deploy some elements that allow controlling the measures implemented in order to assess their effectiveness and to have an insight about them to figure out if there are new problems or there is a need to update their level.

4 Case Study: Adopting the Framework by Ansaldo STS Company

This section shows how the proposed framework is applied at Ansaldo STS Company. Each subsection describes the procedure followed in the goal of adopting it. The seven steps are demonstrated below, showing how they were applied to achieve this overall high level framework of risk analysis and treatment for CPS.

4.1 System Functional Model

The first step is to design a functional model for the system, i.e. it is fundamental to define the scope of the system, the basic components forming the CPS and their composing assets (physical and cyber), and also establishing the relations and dependencies between them. This step is done based on information coming from the owners, since they are familiar and have the knowledge about their system. The functional model will be used to rate the assets against the basic security dimensions Confidentiality, Integrity and Availability (CIA triad), as shown in Fig. 3.

Then provide a high level asset rating for each with the assistance of the system's owners and based on the tables defined below. Figure 4 gives an example of the asset's security dimensions rating, where each asset has a triad rating that represents respectively the confidentiality, integrity and availability rate.

The assets' rating is carried out on each security dimension. Rating represent a pre-valuation step for the assets, where criticality levels will be used with a scale from 1 to 4, where "1" describes the lowest critical level and "4" is the highest. And so, each security dimension gets one of the four levels representing the rate value. For each level, a description is given that helps in choosing the suitable asset's level. The three tables below explain the levels of rating according to each security dimension (Tables 1, 2 and 3).

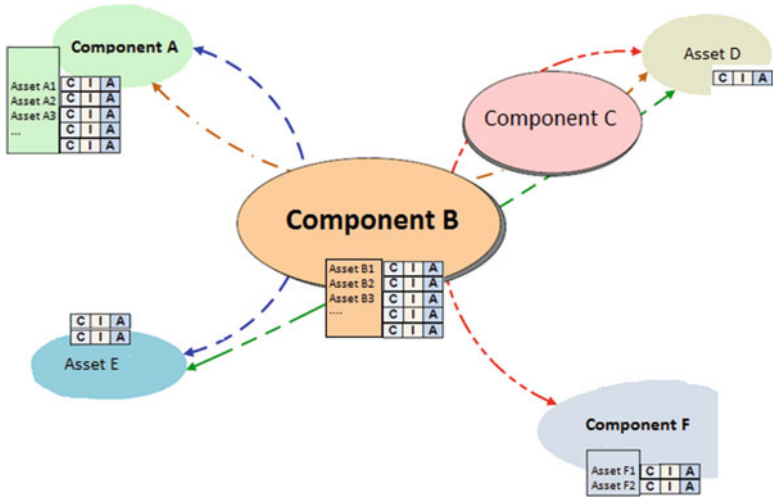
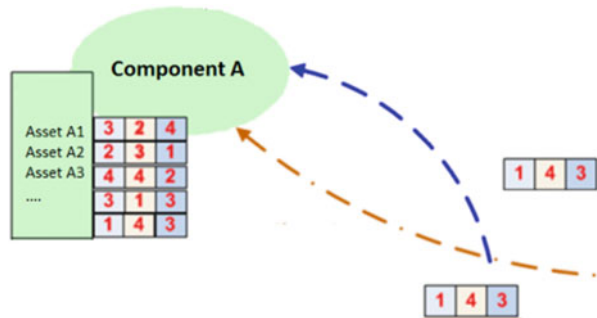


Fig. 3 A functional model example for the CPS

Fig. 4 Rating each security dimension for each asset



4.2 Threat Modeling and Selection: Using RMAT Software

Threat modeling and selection step is about preparing a set of appropriate threats and associate them to asset classes and organizing them also into classes. In particular to execute these actions a dedicated commercial tool, called RMAT, has been identified and adopted. Modeling is meant to prepare the threats selected; RMAT software can be used in the modeling. RMAT is used to create TSV files using a GUI (Fig. 5), a TSV file is a representation for threats. Identifying threats for the TSV file is made by associating threats to asset families. The left panel of Fig. 6 shows the asset families and the threats associated to each one, while the right panel shows the single threats and the asset families associated to each one.

Table 1 Asset’s rating levels for Confidentiality

CONFIDENTIALITY			
Level	Title	Description	Consequence in case of loss of confidentiality
4	Confidential Asset	Asset with a special sensitivity which must be accessed by special authorized staff or services.	Serious impact: Damage could affect directly the system, customer or organizations.
3	Restricted Asset	Assets which must be accessed only by authorized staff members or services.	Significant impact: the reputation of the system can be harmed.
2	Internal Asset	Assets for internal usage in the system which must be accessed only by internal staff.	Negligible Impact: If the confidentiality is breached, small or inconsiderable consequences will happen to the system.
1	Public Assets	Assets of the system which can be accessed by anyone or any service.	Insignificant impact. No damages for the system, customer or organizations.

Table 2 Asset’s rating levels for Integrity

Integrity			
Level	Title	Description	Consequence if there would be an Integrity failure
4	High	The assets must not be compromised by anyone.	Serious impact: The consequences could be catastrophic for the system.
3	Medium	The assets can be compromised by only service personnel with privileged or extended user rights.	Significant impact. The consequences are major and widespread. System errors and services breach persist for a substantial amount of time.
2	Low	The assets can be compromised by internal users even if not having any privileged and extended user right.	Minor Impact. The consequences are noticeable but workaround can be implemented within the system.
1	Negligible	The assets can be compromised by anyone even external users.	Negligible impact. Small or inconsiderable consequences which will not have noticeable influence on the system’s operation.

Table 3 Assets’ rating levels for Availability

AVAILABILITY			
Level	Title	Description	Consequence of Availability deficiency
4	Significant	Unavailability is unacceptable. The asset fails immediately and cannot be re-established by a workaround.	High impact on system’s operation, which may lead to a complete stop or a main impact on the system. Impacts on the public image of the system and/or of the customer.
3	Major	A very short period of unavailability can be accepted during which assets will be unable to provide the intended work.	Medium impact affects the system partially and may lead to a delay in the operation of the system.
2	Minor	A short period of unavailability can be accepted, assets can be re-established by the implementation of alternative procedures.	Small impact on the operation. Small delay with low impact on the operation.
1	Insignificant	Unavailability is acceptable. Asset’s continuity is not affected.	Very-small impact on the operation. No direct delay on the system.

Fig. 5 Creating TSV file using RMAT

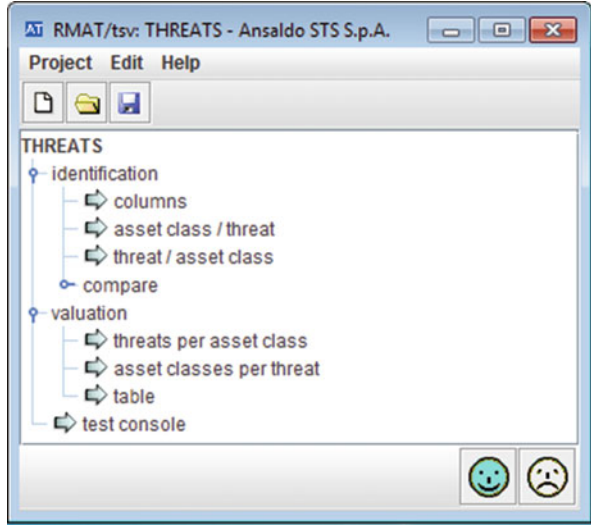
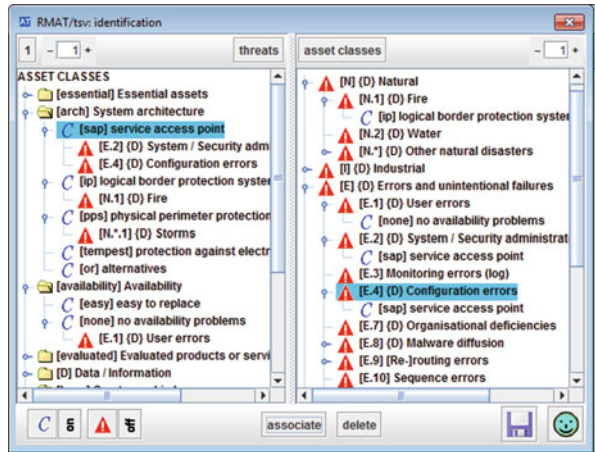


Fig. 6 Associating threats to asset classes using RMAT



The structure of .TSV files that is used to create threat families is:

```

file ::=
    <threat-standard-values>
        { family }0+
    </threat-standard-values>
family ::=
    <family F >
        { threat }0+
    </family>
threat ::=
    <threat Z f [ s ] >
        { set }0+
    </threat>
set ::=
    <set D deg />

```

After creating the appropriate set of threat families, next step is to use it as input to the risk analysis study.

4.3 Conducting Risk Management Study Using MAGERIT Method

For performing this job, Ansaldo STS has identified and adopted a commercial tool, named PILAR, that implements a method called MAGERIT which is suggested by the European Union Agency for Network and Information Security (ENISA). Following a methodical way in a risk management study is significant in order to obtain an efficient study. The objective of MAGERIT method is to cover both risk analysis and treatment for a thorough risk management. MAGERIT is an open methodology for Risk Analysis and Management, developed by the Spanish Ministry of Public Administrations. The purpose of this method is directly related to the generalized use of IT systems, communications, and electronic media. This method follows the international concepts as in ISO 31000 and ISO/IEC 27005 [13]. MAGERIT offers a systematic method for analyzing risks, and helps in describing and planning the appropriate measures for keeping the risks under control. And finally, prepares the organization for the processes of evaluating, auditing, certifying or accrediting, as relevant in each case. On the other hand, PILAR software implements MAGERIT method and is used to perform its steps. Its GUI (graphical user interface) enables the user to execute the MAGERIT method in an understandable and easy way, also making it reproducible. The tool provides fast calculations and generates a quantity of textual and graphical reports. PILAR software has been funded by the Spanish National Security Agency. It is designed to support the risk management process along long periods, providing incremental analysis as the safeguards improve [14]. PILAR enables the user to create a project, identify the assets for the system under study, and generate threats and safeguards and other functionalities (Fig. 7).

Furthermore, PILAR can be customized to use TSV files created by RMAT as input for the risk management study, so in this case the threats will be selected based on the model created before in "Threat Modeling" step.

4.4 Safeguard Implementation

The safeguard implementation step reflects the "DO" phase of the PDCA, which is putting the chosen decisions in the previous treatment plan into operation. At Ansaldo STS, the Defense in Depth (DiD) approach is adopted while implementing safeguards, an approach that is based on layering and that helps in faster detection and slowing down of attacks. In IT environments, DiD is intended to increase the costs of an attack against the organization, by detecting attacks, allowing time to respond to such attacks, and providing layers of defense so that even successful

Fig. 7 PILAR software: homepage

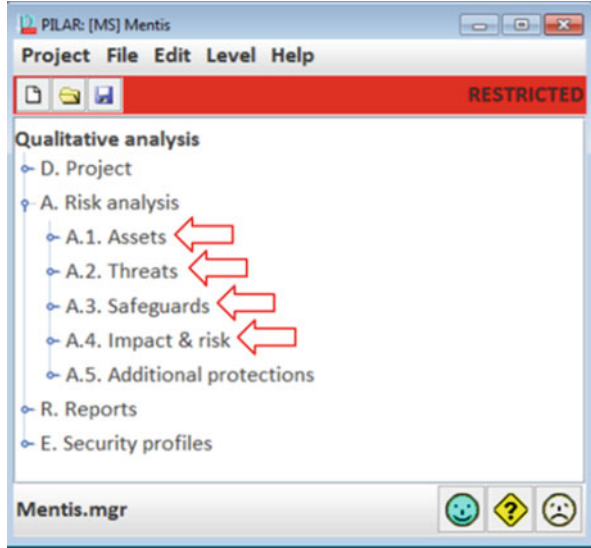
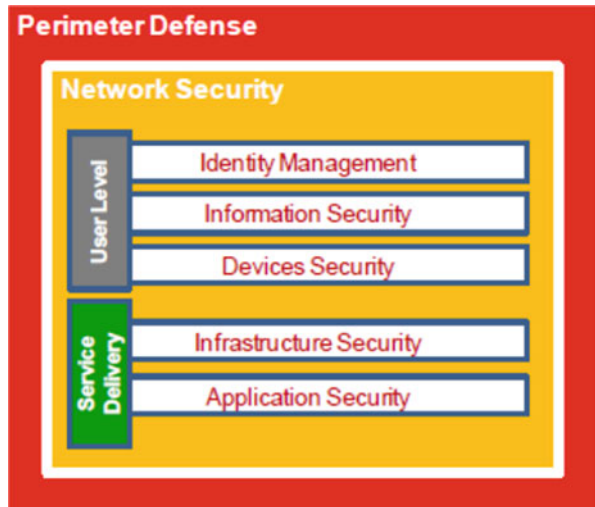


Fig. 8 Layering: defense in depth



attacks will not fully compromise an organization. A DiD strategy is necessary because of the new security threats and the importance of IT security monitoring of assets (Fig. 8).

4.5 *Vulnerability Assessment for Cyber Assets*

The cyber side of a CPS contains various set of assets such as network appliances, servers, software, web applications, databases, etc. At Ansaldo STS, vulnerability assessment is applied basically on 3 levels: operating system, network and application levels.

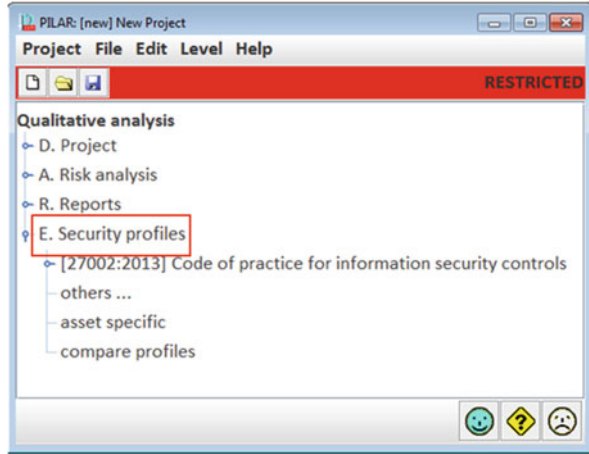
- **OS Vulnerability Assessment:** On the level of operating system, what is meant is to apply host vulnerability assessment through scanning specific hosts. This allows the administrators to go beyond testing for known network vulnerabilities, but also examining more vulnerabilities such as patch levels, check OS configuration, and installed software on computers running operating system.
- **Network Vulnerability Assessment:** Network scanners are useful to analyze the network, and hosts on the network to detect vulnerabilities. Nmap (Network Mapper) is a security scanner used on this level to discover hosts and services on a computer network, thus building a “map” of the network. Nmap features include host discovery, port scanning, OS detection, which all help in finding and exploiting vulnerabilities in the network.
- **Web Application Vulnerability Assessment:** This can be done using automated web application and web services vulnerability scanning solutions that apply attack algorithms and determine the existence and relative severity of vulnerabilities. Some dedicated tools employ an extensive arsenal of attack agents designed to detect security flaws in web-based applications. Such tools probe the system with thousands of HTTP requests and evaluates each individual response. This assessment detects vulnerabilities, pinpoint their location in the application, and recommend corrective actions.

4.6 *Compliance*

Compliance can be oriented to internal policies and rules or to external laws and regulations, but in any case it represents a fundamental step in order to maintain the organization control inside its specific regulatory environment. PILAR software can be also used to conduct this step by using a security profile (EVL file) that is a description for a list of policies that a system would comply to. It is a view over a collection of safeguards that aim to protect a system. Security profiles may focus on some specific aspects, or may be general. The use of a security profile in a project is basically to check and ensure compliance. It is also possible to create custom security profiles, while some widely known are already available e.g.: ISO/IEC 27002. PILAR maps security profiles to its safeguards in such a way to estimate to which extent the system is compliant (Fig. 9).

After loading a security profile into the project, the set of controls for that particular profile are given a score based on the evaluation of safeguards that are

Fig. 9 Applying the security profiles in the compliance step



relevant to those controls only, thus giving a measure to check the compliance of the system to the selected security profile.

4.7 Maintenance and Improvement

At the end, after executing all the steps of the framework, it is critical to monitor and observe if the decisions taken were effective, and if there is a need for maintenance or improvement or even adding a missing measure. On the other hand, in some situations it could be necessary to reduce the cost of a certain countermeasure. Using PILAR in the PLAN phase, the “current” stage represents the current state of the system, and “target” stage represents the goal to reach (Fig. 10). However, now in the “ACT” phase, a new target (Fig. 11) will represent the new goal to achieve based on the new observations and analysis done, and putting all (new) safeguards into operation. The system is monitored and a set of investigations and observations based e.g. on some key performance indicators is done to apply the refinement in case it is required.

5 Conclusion

In recent years, a growth has been seen in the development of various types of Cyber-Physical Systems (CPS). They have brought impacts to almost all aspects of our daily life. Many of such systems are deployed in critical infrastructures, and so, they are exposed to different types of attacks. A Cyber Physical System (CPS) relies basically on information and communication technology, which puts the system’s assets under certain risks especially cyber ones. On the other hand, because of the

Fig. 10 Safeguards values in PLAN phase

current	target
L1	L1-L3
L2	L3
L1	L1
L1	L1-L3
L1	L1-L3
L2	L2
L3	L4

Fig. 11 New Safeguards values in ACT phase

target	new target
L1-L3	L2-L5
L3	L2-L4
L1	L2-L3
L1-L3	L2-L4
L1-L3	L2-L4
L2	L2-L4
L4	L2-L5

characteristics of a CPS, it is more efficient to adopt a solution that is wider than a method, that addresses the type, functionalities and complexity of a CPS. Moreover, following a comprehensive framework ensures a lot of key points such as organizing the steps of a management study, preserving the order of the tasks without missing one, and basically doing the work once in a formalized structure, which is the key spirit of what is called “Comprehensive”, and this should lead automatically to the customer satisfaction and ensuring that the risk management study is complied with laws and regulations. In this chapter, a holistic framework is proposed that breaks the restriction to a traditional risk assessment method, and encompasses wider set of procedures which can be followed in the risk management study for the CPSs, giving more attention to the cyber side that usually controls the physical side of CPSs. Finally, this framework is also ready to accommodate another two security dimensions which are the “authenticity” and “traceability”, that are relevant and should be addressed as security requirements for the risk management of CPSs.

References

1. Peng Y, Lu T, Liu J, Gao Y, Guo X, Xie F (2013) Cyber-physical system risk assessment. Paper presented at ninth International conference on intelligent information hiding and multimedia signal processing
2. Ansaldo STS CBTC communication based train control. <http://www.ansaldo-sts.com/sites/ansaldosts.message-asp.com/files/imce/cbtc.pdf>. Accessed 4 May 2018
3. Chen B et al (2015) Security analysis of urban railway systems: the need for a cyber-physical perspective
4. Andrew F, Emmanouil P, Pasquale M, Chris H, Fabrizio S (2016) Decision support approaches for cyber security investment
5. Ansaldo Signalling and Transportation Systems (Ansaldo STS). <http://www.ansaldo-sts.com/en/about-us/>. Accessed 4 May 2018
6. Balvir S, Amarjeet S (2015) A roadmap to data security of automated university examination system
7. Annual Emerging Cyber Threats Report. Georgia Tech Information Security Center. <http://www.gtisc.gatech.edu/>. Accessed 4 May 2018
8. Internet Security Threats Report. Symantec. <http://www.symantec.com/threatreport/>. Accessed 4 May 2018
9. The CERT guide to insider threats: how to prevent, detect, and respond to theft of critical information, sabotage, and fraud. www.cert.org/archive/pdf/insidercross051105.pdf. Accessed 4 May 2018
10. Hunker J, Probst CW (2011) Insiders and insider threats—an overview of definitions and mitigation techniques. *J Wirel Mob Netw Ubiquitous Comput Depend Appl* 2(1):4–27
11. Mokalled H et al (2017) The importance to manage data protection in the right way: problems and solutions. In: *Optimization and decision science: methodologies and applications*: ODS. Sorrento, Italy, September 4–7, pp 69–82
12. ENISA Threat Landscape Report 2017. 15 top cyber-threats and trends. <https://www.enisa.europa.eu/publications/enisa-threat-landscape-report-2017>. Accessed 4 May 2018
13. MAGERIT – version 3.0. Methodology for information systems risk analysis and management. Book I – The Method, Madrid, July 2014
14. PILAR. Risk analysis and management- help files, version 6.2, August 17, 2016

Part II
Evaluation Methodologies and Tools

Supporting Cybersecurity Compliance Assessment of Industrial Automation and Control System Components



Janusz Górski and Andrzej Wardziński

Abstract The chapter presents a case study demonstrating how security requirements of an Industrial Automation and Control System (IACS) component can be represented in a form of Protection Profile that is based on IEC 62443 standards and how compliance assessment of such component can be supported by explicitly representing a conformity argument in a form based on the OMG SACM metamodel. It is also demonstrated how an advanced argument assessment mechanism based on Dempster-Shafer belief function theory can be used to support assessors while analyzing and assessing the conformity argument related to an IACS component. These demonstrations use a NOR-STA tool for representing, managing and assessment of evidence-based arguments, which have been developed in our research group.

Keywords Cybersecurity · IACS component · Protection profile · Security standards · Evidence-based argument · Conformance case · Certification · Tools

1 Introduction

Cybersecurity assessment of an IACS (Industrial Automation and Control System) component involves identification and examination of its critical assets, related threats and security functions which aim at preventing the threats from occurrence

J. Górski (✉)

Faculty of Electronics, Telecommunications and Informatics, Department of Software Engineering, Gdańsk University of Technology, ul. Narutowicza 11/12, 80-233 Gdańsk, Poland
e-mail: janusz.gorski@pg.edu.pl

A. Wardziński

Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Gdańsk, Poland

Argevide sp. z o.o, Gdańsk, Poland
e-mail: andrzej.wardzinski@pg.edu.pl

and/or from violating security of the assets [1]. For each security function, a set of more detailed security requirements can be specified down to the level where the satisfaction of each requirement can be demonstrated by the available evidence. Specification of critical assets, related threats, security functions and the corresponding security requirements together with the contextual information form what is called a *protection profile*. Protection profile is an implementation-independent set of generic security requirements for a family of components and is usually used as the reference in the security assessment and certification process. It is expected that the manufacturer of a component provides evidence demonstrating that the security requirements specified in the protection profile are met by the component. The assessment of the support given by this evidence to the security requirements is part of the component security certification process.

The evidence comes from different sources, including compliance examination based on the submitted documentation, evidence resulting from security testing, and evidence related to development, shipping, installation and maintenance processes of the component. Figure 1 illustrates how evidence is used in relation to the main elements of the protection profile of a given component.

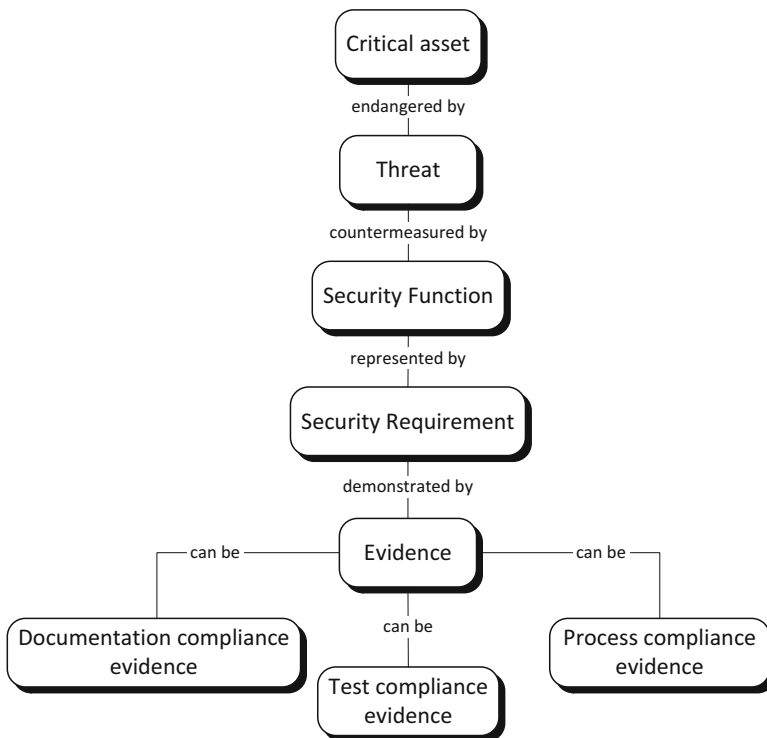


Fig. 1 Evidence-based cybersecurity assessment schema of an IACS component

In this chapter we present the idea that security of a component can be claimed by building an evidence-based argument which argues that the security functions identified in the related protection profile are adequately implemented by the corresponding security requirements, where the satisfaction of security requirements is demonstrated by the available evidence. In general, such an argument represents a *security assurance case* of the considered component (for a meta-model of assurance cases see [2] and for recommendations related to assurance cases see [3]). Different tools are available to support development of assurance cases. In our case study we have used NOR-STA [4] which supports integrated management of argument, evidence and assessment.

Following [1, 5] we assume that a component delivered by its vendor is being evaluated against security requirements which are represented in a protection profile specific for a given family of components. We assume that a mechanism for defining, endorsing and maintaining the protection profiles of IACS components is available to the vendors, users and certification bodies. The following are examples of IACS component families [6]: engineering software, firewall, historian station, manufacturing execution system server, Programmable Logic Controller (PLC), Remote Telecontrol Unit (RTU), SCADA client, SCADA server, switch, VPN gateway, WIFI access point.

In this chapter we first introduce the IEC 62443 concepts to which we refer while describing protection profiles of IACS components and then we introduce our case study – a protection profile of the RTU (Remote Terminal Unit) family of components. Then we present how security requirements of the protection profile were represented in the form of an evidence-based argument pattern (called *conformance template*) and how such conformance template could be used to develop a complete conformance argument of a component belonging to the RTU family. This is followed by a demonstration how the argumentation assessment mechanism based on Dempster-Shafer belief functions theory can be used to support compliance assessment of IACS components against the security requirements. At the end, we summarize our experiences in the conclusions.

We demonstrate our ideas using the NOR-STA system for developing, maintaining and assessing evidence-based arguments [7].

2 Related Works

Protection Profile is one of the core concepts in Common Criteria [5] and together with the concept of Security Target (ST) refers to the security requirements related to the target object subjected to security assessment. These concepts were used (with some modifications) in [1] and we follow [1] in this respect.

Using evidence-based arguments to demonstrate conformity has been argued in [8] and applied in different domains, including medical, oil and gas, automotive and others. Several researchers attempted to demonstrate and assess security by developing explicit assurance cases. [9] proposes an argument structure that

decomposes the main security claim into four sub-claims: (C1) *System security requirements are effectively formulated*, (C2) *System security requirements are captured in design*, (C3) *System implementation is secure*, and (C4) *Operational security requirements compliance measures are clearly defined* (effectively put in place). Here, claim C1 corresponds to a Protection Profile and its argumentation strategy is to identify all threats, possible attack surfaces, attack scenarios and effective counter-measures which are translated into implementation and operation related security requirements.

[10] presents an approach where the goal is to demonstrate a set of security capabilities (like Automatic Logoff, Transmission Confidentiality or Cyber Security Product Upgrades) and an argumentation pattern is used to demonstrate each of these capabilities. This approach has been used in IEC 80001 series of standards (in particular part 2–9 published in 2017 includes guidance for use of security assurance cases to demonstrate device security).

A systematic approach to develop an evidence-based argument demonstrating that security requirements are met has been proposed in [11]. The approach is based on incremental development of the security argument as the design and implementation decisions are made and providing evidence in the development and testing process.

3 Introduction to IEC 62443

IEC 62443 is a series of standards and technical reports addressing security assurance of Industrial Automation and Control Systems. The standards apply to manufacturers, integrators as well as to end-users (the standards were initially developed by the International Society for Automation and they are also referred to as ISA99 standards [12]). IEC 62443 consists of several standards covering four areas: general definitions and metrics, policies and procedures for the plant owners and suppliers, security requirements for systems, and security requirements for components.

In IEC 62443, security requirements for IACS components are decomposed into seven *Foundational Requirements* (FR). These FRs are the categories used to organize technical security controls and form the basis for subsequent more specific requirements. They are as follows [13]:

- FR1: *Identification and authentication control* (IAC): necessary capabilities to reliably identify and authenticate all users (humans, software processes and devices) attempting to access the *Target of Evaluation* (ToE) shall be provided.
- FR2: *Use control* (UC): necessary capabilities to enforce the assigned privileges of an authenticated user (human, software process or device) to perform the requested action on the system or assets and monitor the use of these privileges shall be provided.

FR3: *System integrity* (SI): necessary capabilities to ensure the integrity of the ToE to prevent unauthorized manipulation shall be provided.

FR4: *Data confidentiality* (DC): necessary capabilities to ensure the confidentiality of information on communication channels and in data repositories to prevent unauthorized disclosure shall be provided.

FR5: *Restricted data flow* (RDF): necessary capabilities to segment the control system via zones and conduits (communications channels) to limit the unnecessary flow of data shall be provided.

FR6: *Timely response to events* (TRE): necessary capabilities to respond to security violations by notifying the proper authority, reporting needed evidence of the violation and taking timely corrective actions when incidents are discovered shall be provided.

FR7: *Resource availability* (RA): necessary capabilities to ensure the availability of the control system against the degradation or denial of essential services shall be provided.

For each Foundational Requirement, part 62443-4-2 provides a lists of *Component Requirements* (CR). For instance, the following CRs correspond to FR4 (for the full inventory of CRs see [14]):

CR4.1: Information confidentiality – components need to provide for protection of the confidentiality of information in transit.

CR4.2: Information persistence – components need to provide the capability to erase all information, for which explicit read authorization is supported, from components to be released from active service and/or decommissioned.

CR4.3: Use of cryptography – if cryptography is required, the component needs to use cryptographic security mechanisms according to internationally recognized and proven security practices and recommendations.

4 Case Study: Remote Terminal Unit

In this section we present an excerpt from the Protection Profile of a sample family of IACS components, namely the Remote Terminal Unit (RTU) family. This case study has been elaborated by the National Exercise Team in Poland (NET-PL) while working on the validation of the European IACS components Cybersecurity Certification Framework (ICCF) [1].

4.1 Component Description

Remote Terminal Unit (RTU), in the following text also referred to as Target of Evaluation (ToE), monitors and controls instruments of SCADA systems used in industrial critical infrastructure processes, like oil and gas pipelines, electric

power generation and transmission, chemical manufacturing, physical and technical protection systems, water treatment or others.

RTU main functions include:

- collecting measurements from sensors,
- execution of logic and control calculations,
- user program execution,
- issuing control commands that modify a process,
- communicating with external applications and other devices,
- administration functions to configure or program other functionalities; several administration interfaces are possible: administration console, programming workstation, web-clients,
- supporting removable devices (USB drives, SD memory cards etc.),
- local logging (in particular logging security and administration events),
- remote logging (in particular logging security and administration events).

The usage context of the ToE is presented in Fig. 2. The four parts labelled P1, P2, P3 and P4 shown in Fig. 2 are the interfaces through which RTU interacts with its environment. These parts represent flows (data, control) between RTU and the environment and are included in the scope of security assessment of RTU.

Internally ToE is decomposed into other parts that are relevant from the security perspective. These parts are presented in Fig. 3.

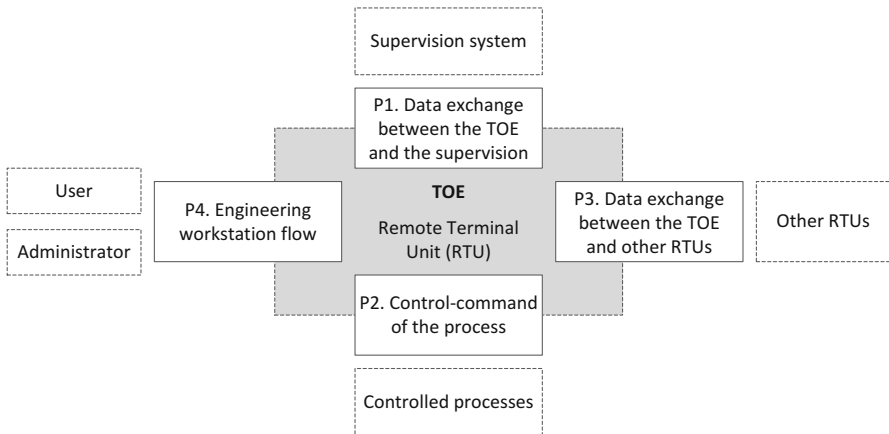
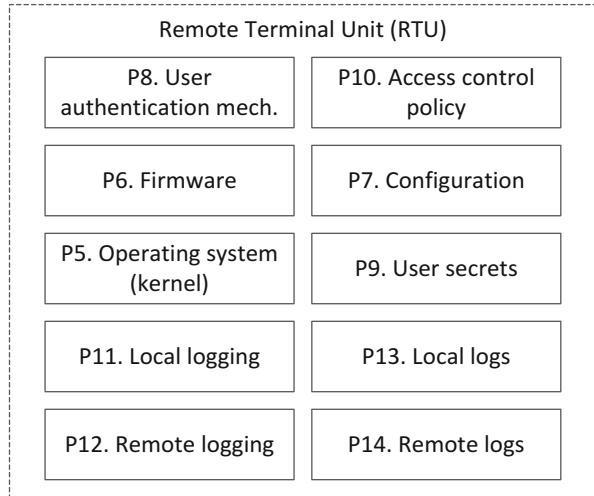


Fig. 2 ToE (RTU) in its target environment

Fig. 3 ToE (RTU) internal structure**Table 1** RTU critical assets (identified by the 'x' symbol)

Part	Security characteristic			
	Availability	Confidentiality	Integrity	Authenticity
P5. Operating system (kernel)			x	x
P6. Firmware		x	x	x
P7. Configuration		x	x	x
P8. User authentication mechanism			x	x
P13. Local logs			x	x
P14. Remote logs			x	x

4.2 Protection Profile of RTU

We assume (based on [1]) the following main elements of a protection profile structure: (1) Description of the family of products (ToE); (2) Parts; (3) Operating conditions; (4) Critical assets; (5) Threats; (6) Protection assumptions; (7) Residual threats; (8) Security functions; (9) Threats vs security functions; (10) Mapping of security functions to security requirements.

RTU parts (presented in Figs. 2 and 3) were subjected to security analysis to assess the risk related to violation of their security properties like availability, confidentiality, integrity or authenticity. The result is a set of *critical assets* that are to be protected against attacks (a critical asset is a security property of a part that needs to be protected by security measures). Selected critical assets of ToE are given in Table 1.

The following are example threats that have been identified as being relevant, during the security analysis of the ToE.

Table 2 RTU critical assets affected by the threats

Threats	Parts				
	P5. Operating system (kernel)	P6. Firmware	P7. Configuration	P13. Local logs	P14. Remote logs
T1. Operating system/firmware alteration	I, Au	I, Au			
T2. Configuration alteration			I, Au		
T3. Local logs alteration				I, Au	
T4. Remote logs alteration					I, Au

- T1. *Operating system/firmware alteration*: The attacker manages to inject and run a corrupted OS/firmware on ToE (for instance, inserts modifications without having the privilege to do so). The code injection may be temporary or permanent and this does include any unexpected or unauthorized code execution. An authorized user may attempt to install a malicious update of ToE by legitimate means.
- T2. *Configuration alteration*: The attacker manages to modify, temporarily or permanently, ToE configuration.
- T3. *Local logs alteration*: The attacker manages to delete or modify a local log entry without being authorized by the access control policy of ToE.
- T4. *Remote logs alteration*: The attacker manages to delete or modify a remote log entry without the receiver (the component hosting the log) being able to notice it.

Table 2 presents which critical assets of ToE can be affected by the identified threats (*Av* stands for availability, *I* for integrity, *C* for confidentiality, and *Au* for authenticity). For instance, integrity of local logs (column P13) can be violated by the *Local logs alteration* threat (row T3).

Critical assets are protected by Foundational Requirements (FR) that are selected to address the related threats. Table 3 presents the selection of FRs to protect the critical assets of RTU.

By grouping the critical assets assigned to the same Foundational Requirements in Table 3, we obtain the list of Security Functions (SF) of RTU. A selected SF of RTU is presented in Table 4.

Then the FRs assigned to a particular Security Function can be decomposed down to the Component Requirements (CR). For instance, consider the assignment of CRs to SF4: *User authentication and authorization of ToE functions*. As shown in Table 4, SF4 has been mapped on two Foundational Requirements: FR1: *Identification and authentication control* and FR4: *Use control*. The present version of 62443-4-2 (still not formally endorsed) maps FR1 on 14 different CRs and FR4 is mapped on 13 different CRs. This leads to the assignment of CRs to SF4 which is presented in Table 5. From Table 4 we see that SF4 addresses only some of the critical assets of ToE and therefore not all CRs will be relevant for these critical assets. In Table 5 this is represented by listing the irrelevant CRs in gray and the relevant CRs in black.

Table 3 Foundational Requirements assigned to critical assets of RTU

Threats	Parts				
	P5. Operating system (kernel)	P6. Firmware	P7. Configuration	P13. Local logs	P14. Remote logs
T1. Operating system/-firmware alteration	Au: FR1, FR2 I: FR3	Au: FR1, FR2 I: FR3			
T2. Configuration alteration			Au: FR1, FR2 I: FR3		
T3. Local logs alteration				Au: FR1, FR2 I: FR3, FR6	
T4. Remote logs alteration					Au: FR1, FR2 I: FR3, FR6

Table 4 Example Security Function of RTU

Security function	Protected critical assets	Foundational requirements	Addressed Threats
<i>SF4. User authorization in TOE functions</i>	Authenticity of: P5. Operating system (kernel) P6. Firmware P7. Configuration P8. User authentication mechanism P13. Local logs P14. Remote logs	FR 1 identification and authentication control FR 2 use control	T1. Operating system/firmware alteration T2. Configuration alteration T3. Authentication violation T4. Local logs alteration T5. Remote logs alteration

5 Support for Representing Security Requirements

Evidence-based arguments are widely used to argue about achievement of some (important) goals. For instance, an argument can justify compliance with a chosen standard or can demonstrate a critical property of a considered object, like safety of a device, security of a service and so on. An argument demonstrating the compliance is called *conformance case* whereas an argument demonstrating the selected property (such as safety, security, reliability, privacy etc.) is called *assurance case*. Recommendations on structuring assurance cases can be found in [2, 3].

Conformance case of a component belonging to a given family of products can be based on a common *argument template* derived from the related protection profile. Such template can be re-used in several concrete arguments [8]. Typically, a template contains the higher (more abstract) part of the argumentation and while

Table 5 Example Security Function with corresponding Component Requirements

Security function	IEC 62443-4-2 requirements
<i>SF4. User authorization in TOE functions</i>	CR 1.1 – Human user identification and authentication
	CR 1.2 – Software process and device identification and authentication
	CR 1.3 – Account management
	CR 1.4 – Identifier management
	CR 1.5 – Authenticator management
	CR 1.6 – Wireless access management
	CR 1.7 – Strength of password-based authentication
	CR 1.8 – Public key infrastructure certificates
	CR 1.9 – Strength of public key authentication
	CR 1.10 – Authenticator feedback
	CR 1.11 – Unsuccessful login attempts
	CR 1.12 – System use notification
	CR 1.13 – Access via untrusted networks
	CR 1.14 – Strength of symmetric key authentication
	CR 2.1 – Authorization enforcement
	CR 2.2 – Wireless use control
	CR 2.3 – Use control for portable and mobile devices
	CR 2.4 – Mobile code
	CR 2.5 – Session lock
	CR 2.6 – Remote session termination
	CR 2.7 – Concurrent session control
	CR 2.8 – Auditable events
	CR 2.9 – Audit storage capacity
	CR 2.10 – Response to audit processing failures
	CR 2.11 – Timestamps
	CR 2.12 – Non-repudiation
	CR 2.13 – Use of physical diagnostic and test interfaces

converting the template to a concrete argument it is necessary to complement it with a more specific argumentation and the supporting evidence. The argument extension explicitly describes strategies of implementing the higher level security requirements in ways that are specific for a particular component. The resulting argument (containing the template, possibly some additional extended argumentation and the supporting evidence) can be subjected to the assessment, as illustrated in Fig. 4.

Templates, concrete arguments, evidence and assessments can be managed with tool support and an example of such tool is NOR-STA [4] which was used in RTU case study. NOR-STA implements TCL metamodel of evidence-based arguments compliant with ISO 15026 [3] and OMG SACM metamodel [2].

5.1 Representing Conformance Arguments

For a given protection profile (PP), the conformance argument demonstrates that all security functions that are relevant for this PP are effective and provide adequate protection of the related critical assets.

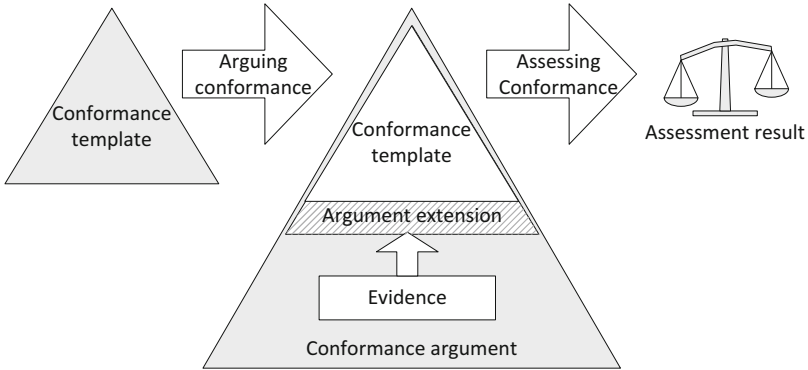


Fig. 4 Use of a conformance template while developing and assessing arguments

- [-] ⓘ **Conformance argument**
 - [-] 🗨️ **ToE satisfies security requirements of RTU PP**
 - [-] 🧠 **Argumentation by demonstrating that the security functions listed in RTU PP are effective**
 - ⚙️ **RTU PP is based on the adequate security risk assessment**
 - + 🗨️ **SF1: Protection of network communication confidentiality**
 - + 🗨️ **SF2: Network communication authenticity and integrity**
 - + 🗨️ **SF3: Protection of critical data confidentiality**
 - + 🗨️ **SF4: User authentication and authorization in TOE functions**
 - + 🗨️ **SF5: TOE functions integrity**
 - + 🗨️ **SF6: Protection of resource availability**

Fig. 5 A fragment of RTU Protection Profile represented in NOR-STA

The following TCL elements are dedicated to representing arguments. Argument conclusion is represented by a *claim* (🗨️) node. A node of type *argumentation strategy* (denoted 🧠) links the claim with the corresponding premises and uses a *rationale* node (denoted ⚙️) to explain and justify the inference leading from the premises to the claim. A premise is a sort of assertion and can be in particular another claim to be further justified by its own premises, a *fact* (denoted 🗨️) represented by an assertion to be demonstrated by the supporting evidence, or an *assumption* (denoted 📄). In addition, the *reference* node (denoted 📄) can be used to point to external documents which are integrated with the argument (for instance, to integrate external files containing evidence supporting argumentation). An auxiliary *information* node (denoted ⓘ) is used to provide more structure and to explain the contents of the argumentation.

Figure 5 illustrates how the above elements are used to represent the topmost structure of the conformance argument for RTU Protection Profile.

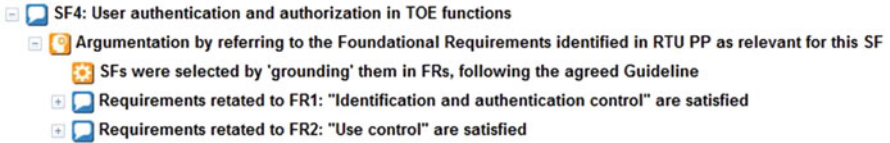


Fig. 6 An argument fragment showing how FRs support SF4 security function

5.2 From Security Functions to Component Requirements

In the RTU Protection Profile, Security Functions are assigned to critical assets and are supported by selected Foundational Requirements, as shown in Table 4. This relationship between Security Functions and the supporting Foundational Requirements is illustrated in Fig. 6 and is specified with the use of an *argumentation strategy* node and the *rationale* node which justifies the inference leading from the premises to the conclusion.

The argumentation strategy and its rationale explain that FR1 and FR2 were identified as the support for SF4 following the Guideline of building Protection Profiles that has been agreed and accepted. Note that if such argumentation strategy is considered acceptable, then to demonstrate SF4 it is sufficient to demonstrate that the requirements related to FR1 and FR2 were satisfied by ToE.

Foundational Requirements (FR) selected to support a given Security Function (SF) can themselves be decomposed down to the level of Component Requirements (CR). For the RTU Protection Profile, an example of such decomposition is presented in Table 5. And this decomposition can be represented by the argument fragment shown in Fig. 7.

In Fig. 7, the same argumentation strategy is used to justify that both FR1 and FR2 will be satisfied if the Component Requirements (CRs) that support particular FR are demonstrated to be satisfied. The strategy is the same for both FRs but it has to be accepted separately. If we accept the argumentation strategy for FR1, the requirement will be satisfied depending on the satisfaction of the component requirements CR1.1, CR1.2, CR1.3, CR1.4, CR1.5, CR1.11 and CR1.12.

Satisfaction of each Component Requirement (CR) can then be argued by referring to some facts that assert about component design, test results, reviews/inspections, handling procedures and so on. Figure 8 illustrates how the satisfaction of CR 1.11 from Fig. 7 could be argued by referring to the recommended best practices of unsuccessful login handling. The facts shown in Fig. 8 are supported by the evidence which can be accessed through the corresponding reference nodes. The files containing the evidence can be stored in any external repository, for instance in the design documentation repository, test results repository and others. For instance, the evidence demonstrating that F1.11.3: *the mechanism for setting limit for unsuccessful logins* is in place could be composed of two pieces of evidence: E1.11.3.1: *an excerpt from the design documentation explaining the mechanism* and E1.11.3.2: *the report from tests verifying that the mechanism works as expected* (see Fig. 8).

- [-] [🗨️] SF4: User authentication and authorization in TOE functions
 - [-] [🔗] Argumentation by referring to the Foundational Requirements identified in RTU PP as relevant for this SF
 - [⚙️] SFs were selected by 'grounding' them in FRs, following the agreed Guideline
 - [-] [🗨️] Requirements related to FR1: "Identification and authentication control" are satisfied
 - [-] [🔗] Argumentation by referring to the related component requirements from IEC 62443-4-2
 - [⚙️] IEC 62443 is an internationally recognized standard supporting selection of security requirements for IACS components
 - [🗨️] CR 1.1: Human user identification and authentication
 - [🗨️] CR 1.2: Software process and device identification and authentication
 - [🗨️] CR 1.3: Account management
 - [🗨️] CR 1.4: Identifier management
 - [🗨️] CR 1.5: Authenticator management
 - [🗨️] CR 1.11: Unsuccessful login attempts
 - [🗨️] CR 1.12: System use notification
 - [-] [🗨️] Requirements related to FR2: "Use control" are satisfied
 - [-] [🔗] Argumentation by referring to the related component requirements from IEC 62443-4-2
 - [⚙️] IEC 62443 is an internationally recognized standard supporting selection of security requirements for IACS components
 - [🗨️] CR 2.1: Authorization enforcement
 - [🗨️] CR 2.3: Use control for portable and mobile devices
 - [🗨️] CR 2.5: Session lock
 - [🗨️] CR 2.6: Remote session termination
 - [🗨️] CR 2.7: Concurrent session control
 - [🗨️] CR 2.8: Auditable events
 - [🗨️] CR 2.9: Audit storage capacity
 - [🗨️] CR 2.10: Response to audit processing failures
 - [🗨️] CR 2.11: Timestamps
 - [🗨️] CR 2.12: Non-repudiation
 - [🗨️] CR 2.13: Use of physical diagnostic and test interfaces

Fig. 7 SF4 supported by FRs and corresponding CRs

- [-] [🗨️] CR 1.11: Unsuccessful login attempts
 - [-] [🔗] Argumentation by referring to the best practices recommendations
 - [⚙️] Best practices represent proven protection mechanisms
 - [-] [🗨️] F1.11.1: Password expiration settings management
 - [🗨️] E1.11.1.1: Design documentation explaining the password expiration mechanism
 - [-] [🗨️] F1.11.2: Checking and handling login errors
 - [🗨️] E1.11.2.1: Design documentation explaining the mechanism for login errors handling
 - [-] [🗨️] F1.11.3: Setting limit for unsuccessful logins
 - [🗨️] E1.11.3.1: Design documentation explaining the limit of unsuccessful logins
 - [🗨️] E1.11.3.2: Report from tests addressing the limit of unsuccessful logins

Fig. 8 Argument fragment how CR1.11 is supported by facts and evidence

6 Support for Conformance Assessment

Conformance arguments can be extended with the assessment data as presented in Fig. 4. In the RTU case study we used the assessment method based on Dempster-Shafer theory of evidence (the details can be found in [15]). The assessment process is explained below.

The assessor issues her/his opinion related to the acceptance/rejection of the assessed object and specifies the confidence level associated with this opinion. The assessed objects are argumentation strategies (in this case the assessor decides if he/she accepts the strategy) and facts (in this case the assessor decides to which extent a given fact has been demonstrated by the evidence supporting this fact). The assessments are expressed using linguistic values. The following values are used to express the decision of the assessor: *acceptable*, *tolerable*, *opposable*, *rejectable*, where *acceptable* means that the evidence fully demonstrates the assessed fact, whereas *rejectable* means that the presented evidence demonstrates the opposite. In addition, the assessor expresses confidence in her/his decision using the following linguistic values: *for_sure*, *with_very_high_confidence*, *with_high_confidence*, *with_low_confidence*, *with_very_low_confidence*, *lack_of_confidence*. In this case, *for_sure* means that the assessor is fully confident in the decision, whereas *lack_of_confidence* means that he/she is fully uncertain (and in this case the decision is irrelevant). The aggregation functions of the mechanism provide for automatic propagation of these assessments into the assessments of the claims of the argumentation (for full explanation of this mechanism see [15]).

The assessment scale can be presented as an *assessment triangle* as illustrated in Fig. 9. The assessment result is a point on the assessment scale (a small shallow token shown at the top of Fig. 9). The scale values are described on the bottom and on the right of the assessment triangle.

Figure 10 presents the result of assessing (a fragment of) the Protection Profile of RTU. The assessed element is fact F1.11.3 marked on the argument element tree. This fact is supported by evidence E1.11.3.1 and E1.11.3.2. The assessor should inspect these evidence items to decide how they support the fact and then should express his/her opinion using the assessment scale. If the evidence is not complete

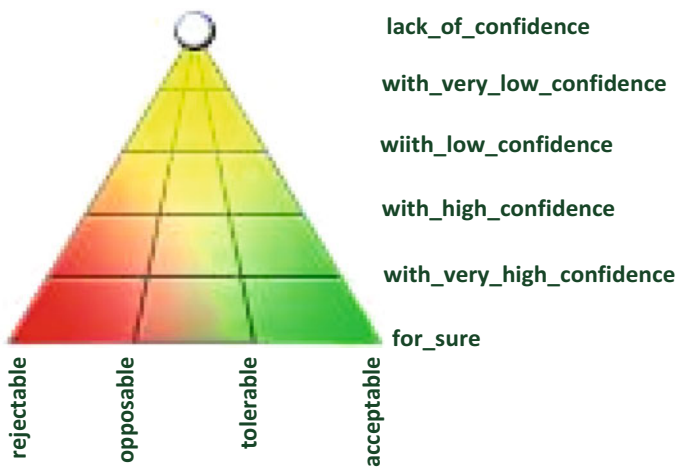


Fig. 9 The assessment scale for Dempster-Shafer method

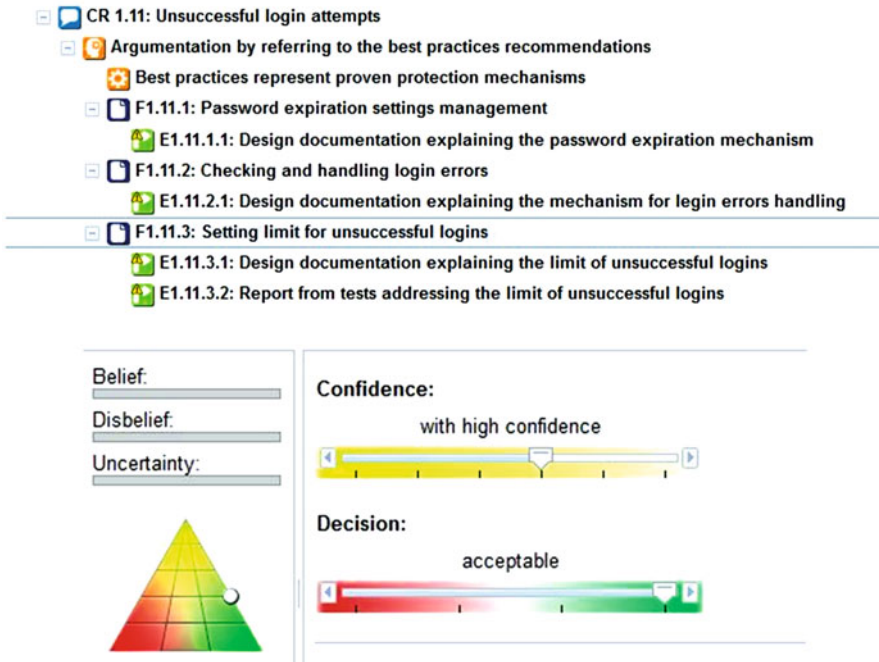


Fig. 10 Assessment of the fact F1.11.3

or ambiguous he/she may give assessment with low level of confidence. Depending on the content of the evidence the result may be acceptance or rejection.

Assume now that the assessor has already issued his/her assessments for facts F1.11.1, F1.11.2 and F1.11.3 and in addition she/he has fully accepted the rationale of argumentation strategy for claim CR1.11. Now facts F1.11.1 and F1.11.2 are fully accepted and fact F1.11.3 has been assessed as “acceptable with high confidence” (as in Fig. 10). In such case the assessment of claim CR1.11 will be calculated automatically from the assessments of the related argumentation strategy and its premises. And the resulting assessment of CR1.11 is “acceptable with very high confidence” as shown in Fig. 11. The assessment results can also be presented with color scale: green, red and yellow colors to represent respectively acceptance, rejection and uncertainty (the colors are not visible in the black and white text of this chapter).

To assess the whole conformance argument presented in Fig. 5 it would be necessary to assess all argumentation strategies and to assess all facts supporting the claims related to Security Functions of RTU. Note however that all those assessments are ‘local’ in the sense that they require that the assessor focuses on just one section of the argumentation structure (when assessing an argumentation strategy, the assessment scope covers the claim, the strategy and premises supporting it). Then, the assessment of the top claim can be calculated automatically by applying assessment aggregation rules.

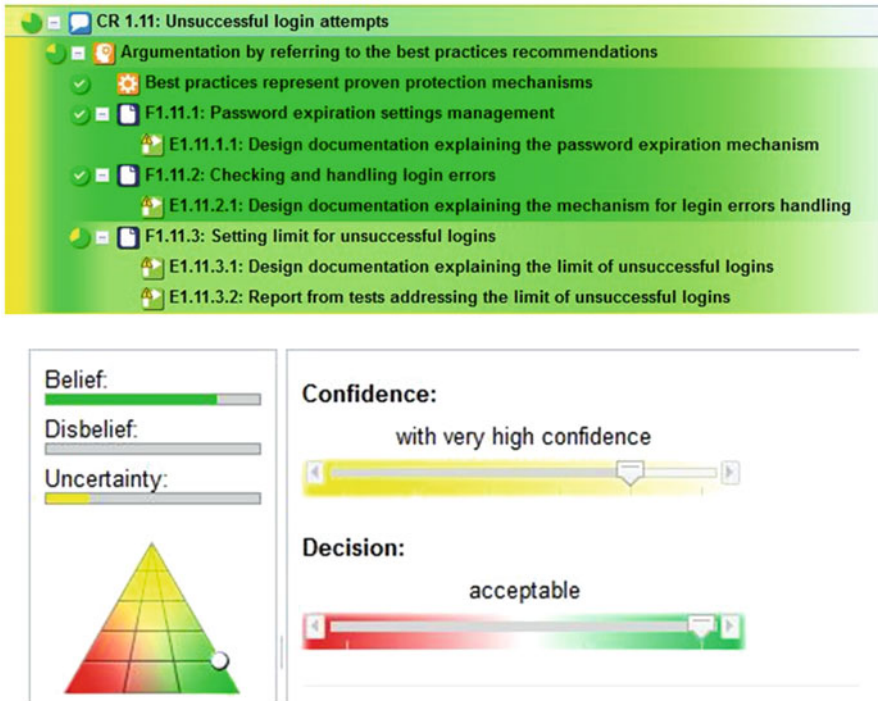


Fig. 11 Assessment of CR1.11 based on the assessments of F1.11.1, F1.11.2 and F1.11.3

7 Conclusions

In this chapter we presented a case study demonstrating how security requirements of an IEC 62443 based Protection Profile of a family of IACS components can be represented in the form of an evidence-based argument and how such argument could be used to support assessment of the compliance of an IACS component.

The proposed approach has the following advantages:

- The security requirements of the Protection Profile can be represented as an argumentation scheme (called conformance template) which can be reused for different components belonging to the same family;
- The template becomes a complete conformance argument by extending it with the argumentation that is specific for a given component and then submitting the evidence and integrating it with the argument;
- Assessment of a conformance argument can be supported by using advanced methods which provide for explicit representation of assessor’s decisions as well as the uncertainty associated with these decisions;
- Automatic aggregation functions facilitate assessment of large arguments which can be encountered in practice reducing the tedious effort of calculating the

overall assessment result and tracking relations between the evidence, security requirements, security functions and objectives;

- The process of template development, argument instantiation for a specific component, integration of the evidence, security assessment, and reporting and visualization of the results can be supported by a dedicated tool.

The protection profiles, related conformance arguments, the evidence supporting the argumentation and the results of conformance assessment form together a complex set of interrelated data and documentation. To process such data accurately and efficiently and to provide for scalability of such processing, it is essential to have an adequate tool support. It is important that such tools allow for seamless cooperation between security experts, component engineers and assessors.

Acknowledgement This work was partially supported by a Statutory Grant of Polish Ministry of Science and Higher Education. The RTU Protection Profile presented in this chapter is based on the RTU Protection Profile originally introduced by Mr. Tomasz Szala from the Mikronika company to the NET-PL group working on validation of the IACS Components Cybersecurity Certification Framework (ICCF).

References

1. Paul Theron Introduction to the European IACS components Cybersecurity Certification Framework (ICCF). DOI:10.276D/717569
2. Structured Assurance Case Metamodel (SACM), version 2.0, Object Management Group (2017)
3. ISO/IEC 15026 Systems and software engineering – systems and software assurance
4. www.aragevide.com/services/en/support/nor-sta/manual (visited 10.10.2017)
5. ISO 15408 (2009) Information technology – Security techniques – evaluation criteria for IT security – Part 1: introduction and general model. ISO
6. <http://www.ssi.gouv.fr/entreprise/guide/profils-de-Protection-pour-les-systemes-industriels/> (visited 7.09.2017)
7. www.aragevide.com
8. Cyra L, Górski J (2011) SCF – a framework supporting achieving and assessing conformity with standards. *Comput Stand Interfaces Elsevier* 33:80–95
9. Ray A, Cleaveland R (2015) Security assurance cases for medical cyber-physical systems. *IEEE Des Test* 32(5):56–65
10. Finnegan A, Mccaffery F (2014) A security argument pattern for medical device assurance cases, In: 2014 IEEE International symposium on software reliability engineering workshops. *IEEE*, pp 220–225
11. Othmane L, Angin P, Bhargava B (2014), Using assurance cases to develop iteratively security features using scrum. In: 2014 Ninth international conference on availability, reliability and security (ARES), *IEEE*
12. International Society of Automation (ISA), www.isa.org (visited 10.08.2017)
13. IEC 62443-1-1 (2009) Industrial communication networks – Network and system security – Part 1-1: terminology, concepts and models, *IEC*
14. IEC 62443-4-2 Technical security requirements for IACS components
15. Cyra L, Górski J (2011) Support for argument structures review and assessment, reliability engineering and system safety, vol 96. Elsevier, pp 26–37

Quantitative Evaluation of the Efficacy of Defence-in-Depth in Critical Infrastructures



Oleksandr Netkachov, Peter Popov, and Kizito Salako

Abstract This chapter reports on a model-based approach to assessing cyber-risks in a *cyber-physical system* (CPS), such as power-transmission systems. We demonstrate that quantitative cyber-risk assessment, despite its inherent difficulties, is feasible. In this regard: (i) we give experimental evidence (using Monte-Carlo simulation) showing that the losses from a *specific* cyber-attack type can be established accurately using an abstract model of cyber-attacks – a model constructed without taking into account the details of the specific attack used in the study; (ii) we establish the benefits from deploying *defence-in-depth* (DiD) against failures and cyber-attacks for two types of attackers: (a) an attacker unaware of the nature of DiD, and (b) an attacker who knows in detail the DiD they face in a particular deployment, and launches attacks sufficient to defeat DiD. This study provides some insight into the benefits of combining design-diversity – to harden some of the protection devices in a CPS – with periodic “proactive recovery” of protection devices. The results are discussed in the context of making evidence-based decisions about maximising the benefits from DiD in a particular CPS.

Keywords Stochastic models · Defence-in-depth · Power transmission system · Adversary model · Cyber-attacks · NORDIC-32 · IEC 61850

1 Introduction

Security of *industrial control systems* (ICS) used to control *critical infrastructure* (CI) has attracted the attention of researchers and practitioners. The evidence is overwhelming that the services offered by CI are somewhat robust with respect to single component failures of the underlying network. The reaction to multiple

O. Netkachov · P. Popov (✉) · K. Salako
University of London, London, UK
e-mail: Oleksandr.Netkachov@city.ac.uk; P.T.Popov@city.ac.uk; K.O.Salako@city.ac.uk

© Springer International Publishing AG, part of Springer Nature 2019
F. Flammini (ed.), *Resilience of Cyber-Physical Systems*,
Advanced Sciences and Technologies for Security Applications,
https://doi.org/10.1007/978-3-319-95597-1_5

and cascade failures, however, is much more difficult to understand and to predict, especially when *cyber-attacks* are taken into consideration.

Dealing adequately with cyber-threats requires a credible assessment of the effectiveness of cyber-security controls deployed in a particular system. This is particularly important if the results from the analysis are used to support *decision making*, e.g. about how to maximize the benefits from a given limited investment. Cyber-security *assessment* has matured over the last decade.¹ Yet, recommendations to deploy specific security controls are often made with *no quantification* of the benefits these are likely to bring to a *particular system*. The assessment results, especially when qualitative assessment techniques are used, are often difficult to reproduce. Decision makers struggle to justifiably answer practical questions such as “How much should I invest in improving cyber-security?”, “How much better is my system after spending the available budget on additional cyber-security controls?”, and “Have I done enough to secure my system?”

Probabilistic models for assessment are widely used in critical systems, for quantitative reliability assessment as well as highlighting *serious misconceptions*, e.g. the well-known controversy surrounding the quantification of the benefits from design diversity for software reliability [1, 2]. The success of these models motivated the present work – using a similar style of modelling, we develop a method for cost-benefit analysis of defence-in-depth in a CI. *Defence-in-depth* (DiD) – a multi-layered approach to defending against accidental and design faults – has been widely used in safety-critical systems for many decades. The essence of DiD is that a number of defence mechanisms, typically diverse in nature, are deployed to defend a system from threats, such as accidental/design faults or malicious activities (e.g. cyber-attacks). Respectable bodies, e.g. ICS-CERT, have recommended DiD for cyber-security of ICS [3]. While DiD has been demonstrated to bring significant *safety* benefits in safety-critical systems, its benefits with respect to cyber-threats are yet to be demonstrated convincingly. In this chapter we take some steps in this direction.

In this chapter:

- We study how the behaviour of a complex system model (of a power transmission system) is affected by the level of abstraction in modelling the effect of cyber-attacks on smart devices (i.e. those devices containing non-trivial software) deployed in a power transmission network. We compare the behaviour of the same system model using two alternative models for the effect of cyber-attacks on the smart devices: i) a *conceptual (i.e. abstract) model* of the reliability of smart devices deployed in adverse environments and ii) a more detailed model of the effects of successful, specific cyber-attacks described in our previous work [4]. Our results demonstrate how the abstract cyber-attack model *can be tuned by a suitable parameterisation*, so that the system model behaves *comparably* to how it behaves using the more detailed cyber-attack model. This observation

¹A range of standards deal with risk assessment including cyber-attacks on industrial control systems, e.g. IEC 62443, ISO/IEC 15408, ISO 27005, etc.

suggests that a model-based risk assessment (or a cost-benefit analysis) can be performed, perhaps even for *unknown cyber-threats*, by using an abstract model of cyber-attacks with a suitable parameterisation.

- We apply the abstract model in studying the benefits from deploying a specific form of DiD in a power transmission network and its respective ICS. Here, DiD involves replicating some of the smart devices using *design diversity*, e.g. devices from different vendors are combined, together with a maintenance policy, such as “proactive recovery” [5] or “cleansing” [6].

The rest of the chapter is organized as follows: In Sect. 2 we state the problem of quantifying the benefits from DiD against cyber-attacks in CI. In Sect. 3 we provide a brief description of the case study and the particular models adopted for DiD. Section 4 summarizes our findings, which we discuss in Sect. 5. Related research is highlighted in Sect. 6 and, finally, Sect. 7 concludes the chapter with an outline of directions for future research.

2 Problem Statement

Investment in improving CI resilience is high on the agenda of many companies’ boards. An investment decision is typically taken in the face of a large number of alternatives and uncertainties, thus requiring an evaluation and comparison of the efficacy and associated risks of employing each of these alternatives. An example investment decision, taken from a power systems context, considers whether or not the monitoring of physical network assets should be accomplished using either *Wide Area Measurements Systems* (WAMS) that employ high frequency GPS-synchronized *phasor-measurement units* (PMU), or a more traditional monitoring network comprised of low frequency *remote terminal units* (RTU).

WAMS, being newer technology than RTU-based monitoring solutions, allow operators in control centres to conduct more sophisticated and accurate calculations of a power system’s state. However, the adoption of WAMS appears to be largely driven by the “obvious” superiority of the new WAMS technology over more traditional RTU-based state estimators.² But there are risks in adopting this new technology, not least its apparent sensitivity to cyber-threats, and such risks do not appear to have been adequately addressed. For instance, take WAMS dependence on GPS. A recognized concern, GPS signal jamming, is a critical failure-mode. Recent studies, e.g [7, 8], demonstrate that sophisticated *man-in-the-middle* (MiM) attacks on PMU readings may remain unnoticed by WAMS state-estimation algorithms and lead to significant biases in the power system state perceived by operators in control centres. This approach to adopting technological improvements – based on

²This came to our attention from private conversations with WAMS vendors.

“obvious” benefits and either disregarding or underestimating new risks – seems to follow a well-established pattern in industry.³

Furthermore, while WAMS may well bring benefits to both vendors and adopters alike in the long-run, short term risks for *early adopters* may be significant; quantification of these risks seems highly desirable. The decision “to invest now or not” should be based on a sound *cost-benefit analysis*. Some of the costs and benefits are clear: (i) technical advantages of WAMS over traditional state-estimators are demonstrable; (ii) the upfront costs are known. The costs due to failures, however, are much more difficult to estimate. They depend on the frequency of failures and on the harm these failures cause, how good the new technology is in the face of failure, the particular operational environment, and any *additional controls* used in the particular deployment. For instance, in some installations, the WAMS dependence on GPS may be compensated by deploying *atomic clocks* which allow the PMU to continue to work with accurate timestamps even if the GPS signal becomes unavailable (e.g. due to accidental failure of a GPS receiver or due to jamming). Absence of atomic clocks in a particular installation will make WAMS dependence on GPS a serious risk for this system. Similarly, if controls are in place (e.g. strong encryption) which make MiM attacks unlikely, perturbation of the PMU readings may be assumed unlikely, which in turn will justify ignoring the problems discussed in [7, 8]. Finally, if one is uncertain about the quality of the controls,⁴ then a more detailed study of how the quality of a particular control impacts system operation may be needed.

Given all of the foregoing, we contend that a sound risk assessment (or a cost-benefit analysis) should be done for a *specific system*, rather than solely relying on the results of pilot studies conducted elsewhere or merely adopting generic “best practices” which may ignore important deficiencies of a specific system. Consider the case when one needs to spend a fixed budget, sensibly, to improve a particular system. A rational approach to solving this problem would be exploring the space of possible system changes, i.e. consider a number of alternative ways of investing in CI resilience and ranking the alternatives according to the benefits each of these bring. It is typically too expensive for more than a few alternatives to be tried for real. But this problem can be overcome by *using high fidelity models* – one per plausible alternative – and conducting a model-based comparison. Provided the models are credible [10], one can establish the losses due to failures under comparable threat scenarios, an essential consideration for making a sound cost-benefit analysis of the planned investment.

³In a highly regarded book on the theory of “disruptive innovation”, [9], Christensen demonstrated that the initial technological inferiority of products and services is typically temporary and is no impediment for adopting disruptive products/technologies, provided this addresses real market needs (e.g. creates new markets, reduces the cost, etc.). Here, the WAMS technology may be inferior in terms of cyber-risks, e.g. GPS jamming is not a problem at all for low-frequency state-estimation, but it *is* a critical failure mode for WAMS.

⁴For instance, strong encryption may guarantee the integrity of PMU readings, but i) the use of encryption in practice is not guaranteed, and ii) encryption keys may be compromised.

Such an approach is feasible – we study the benefits from adopting a specific form of defence-in-depth on a non-trivial CPS such as NORDIC-32, a reference architecture of a power transmission system. Of particular interest is the effect of hardening the instrumentation/control by introducing *design diversity* in the protection devices of power lines, generators and loads. We consider investment in protection devices by replacing legacy devices with fault-tolerant, two-channel protection devices, each of which works as a 1-out-of-2 system. That is, the specific function of the device (e.g. a line protection) only fails if both channels become failed simultaneously. We assume that the channels, although functionally equivalent, are implemented differently. For example, when protection is based on different algorithms (functional diversity) or on different implementations (by different vendors) of the same algorithm. There are two important consequences of such *diversification*: the channels may be less likely to i) fail simultaneously due to *the same design fault* (e.g. the same software bug); ii) contain *the same exploitable vulnerabilities* (than if the channels were identical). Thus, repeating the same attack on each of the channels is unlikely to compromise both. Compromising both channels is still possible, but may require different attacks be carried out either simultaneously (or in quick succession, but with a very short duration between each attack) or at different times.

3 The Case Study

We use a non-trivial case study of a power transmission network, NORDIC-32, to demonstrate our approach. The system model was developed by the FP7 EU project AFTER – the NORDIC-32 network was enhanced with an industrial distributed control system (IDCS), compliant with the international standard IEC 61850 “Communication networks and subsystems in sub-stations”. A detailed description of the system model is beyond the scope of this chapter, but a short summary is provided below.

3.1 The Cyber-Physical System under Study

The transmission network (Fig. 1) consists of a large number of transmission lines, which connect 19 power generators and 19 loads. All of the connections of the lines, generators and links are done in 32 substations.

Each substation is arranged in a number of bays. Each bay is responsible for connecting a single element – a line, a generator or a load – to the transmission network. The substations are assumed compliant with IEC 61850. Figure 2 shows an example of one such substation (substation 4011). The other substations have similar architecture, but they may differ in their number and types of *bays*. Some substations have generators and/or loads, and all sub-stations contain Line-bays connecting transmission lines to the bus-bar of the particular substation.

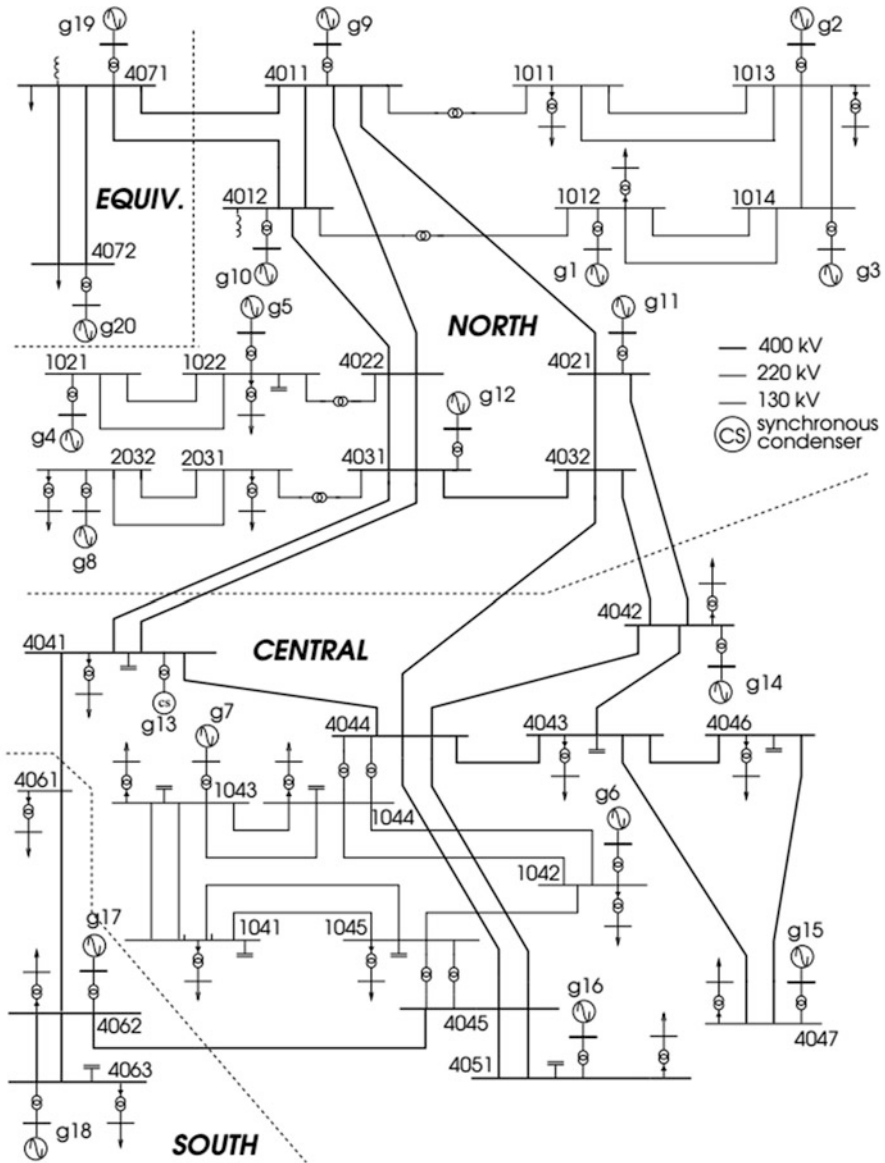


Fig. 1 NORDIC-32 power system topology. This topology is well documented in the technical literature, e.g [11] or the more easily accessible [12]

Each sub-station has a *Local Area Network (LAN)*, allowing local devices to communicate with one another. The LAN is protected from the rest of the world by a *firewall*. Legitimate traffic in and out of the sub-station is allowed, of course.

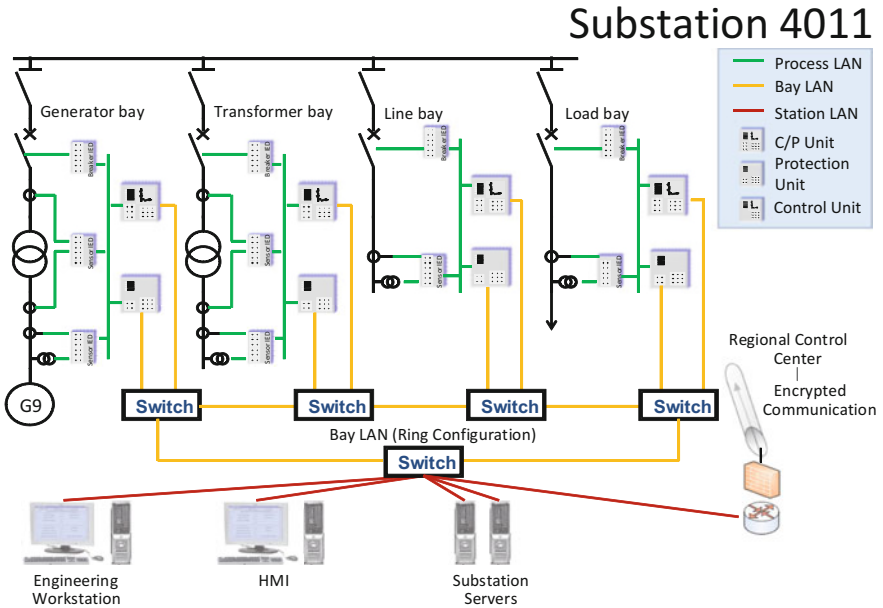


Fig. 2 Substation topology (IEC 61850 compliant)

The substations are connected via a sophisticated communication infrastructure (Fig. 3), which includes a number of control centres, communication channels and data centres.

During system operation, each protection/control function (with respect to the individual bays in substations) that is needed to maintain system integrity is either available when needed or unavailable. Availability is determined by the state (operational or not) of the equipment required for enacting the control function. For instance, shedding off a specific load to balance the power between the available generators and loads in the system, can only be achieved if the respective components – relays, communication from control centres to the respective substations, etc. – are all operational. So, in a model of the system, availability of a control function is determined by a *predicate* on the minimal cut set for the function (measurement, protection or control). Only when the predicate evaluates to “true” is the respective function available; if the predicate evaluates to “false” instead, the respective function becomes unavailable. The function will only deliver the expected outcome when it is available. If an unavailable function is called upon to execute, it will fail to achieve the required outcome. For instance, if the function to shed some load is called upon when it is unavailable, the load will not be disconnected from the power network.

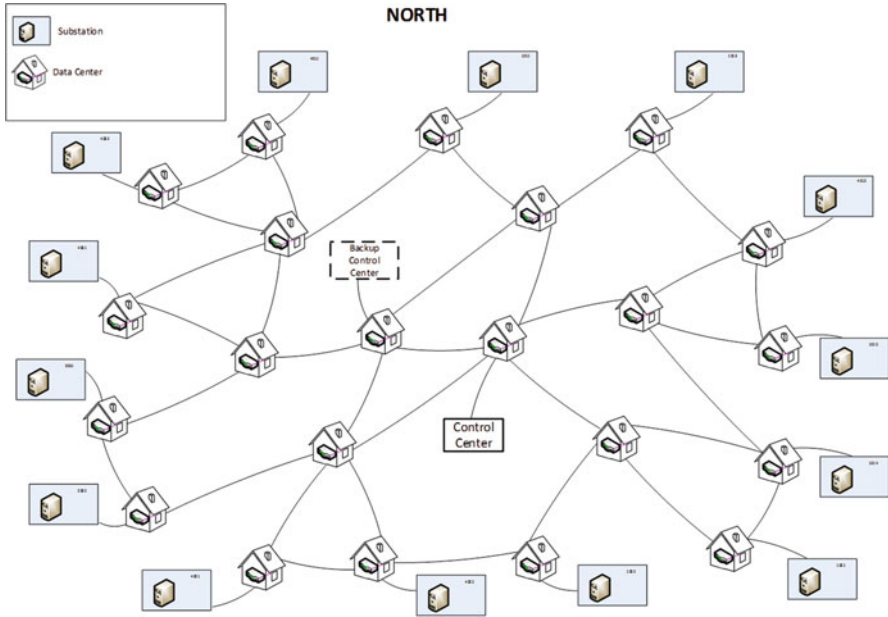


Fig. 3 Communication topology (EMS + SCADA)

Each bay is responsible for (dis)connecting one element from the transmission network. *Protection devices* (breakers) serve to disconnect power elements from the transmission network, e.g. because of the over-loading of a line or a generator. *Control devices*, on the other hand, are used to connect or disconnect power elements from the network and are typically used by either the operators in the respective control centres or by *special purpose software* (SPS) designed to undertake some of the operators' functions automatically.

Some functions are implemented using *functionally redundant components*, others are not.

We model the behaviour of the entire system using a hybrid model: a combination of probabilistic and deterministic models to capture different aspects of system behaviour. Each element in the system model – whether it be a power element or an element for instrumentation and control – is modelled as a *stochastic state machine*. The effect of component failures on power-flow across the network is captured by a deterministic power-flow model (a DC approximation). A more detailed description of the method used to create the system model is given in [13]. Each state machine has at least two states – “OK” and “Failed”. Some state machines (e.g. all power elements) may have an additional state, e.g. “Disconnected”. Some other components containing software, such as those that facilitate control and instrumentation, may have the additional state “Compromised”, the semantic of which we detail later.

Depending on the element type, its model, in addition to a state machine, may include specific additional *properties* needed to capture the functionality of the component. For instance, the model of a generator will have a property defining the maximum power that the generator can produce; the model of a load will have an additional property defining the power consumed; a power line will have two properties – one that defines the line capacity and another that defines the power (current) through the line. The full list of properties for the different components that occur in the system model are beyond the scope of this chapter. However, we provide an illustration of these concepts in Appendix A. The listed properties for a given element may vary depending on the level of detail used in the model – e.g. depending on whether DC or AC power-flow calculations are used to establish the new state of the power network following a disruption. The complete model used in the study can be found in [14].

The reader familiar with state-based probabilistic models may have realised that these properties extend the state space of the state machines *implicitly*. This complication is handled in our model by having a clear separation between the state captured by the states of the respective state machine and the state extension captured by the values of the various properties. The discrete part of the state evolves according to the logic built into the topologies of the state machines of the individual components. The dependencies between the state-machines are captured by different models, chief among them power-flow calculations that use the state of components across the entire power network, and predicates that determine the operational status of protection/control/measurement devices based on the state of all related components, etc. This pragmatic approach allows us to apply in the studies well-known methods of solving Markov processes despite the use of properties attached to some of the components.

3.2 *Modelling Protection Devices*

In our study we compare the behaviour of systems using non-replicated protection devices, with the behaviour of systems with (some) replicated protection devices – replication being the use of functionally equivalent, but “diverse”, channels in the devices.

The state machines of both 1-channel and 2-channel protection devices are shown in Figs. 4 and 6, respectively. In these diagrams we refer to a “Compromised” state which will be defined in detail in Sect. 3.3.

A Markov chain is shown in Fig. 5, which corresponds to the state-machine shown in Fig. 4 under the assumption of exponentially distributed sojourn-times.

The states of the Markov chain correspond to the states in the UML diagram, except for two differences: (i) there are two failed states – “Failed” and “Failed Compromised” states. This is necessary since the state machine in Fig. 4 models a non-Markov process: which state the device is restored to from a failure is dependent

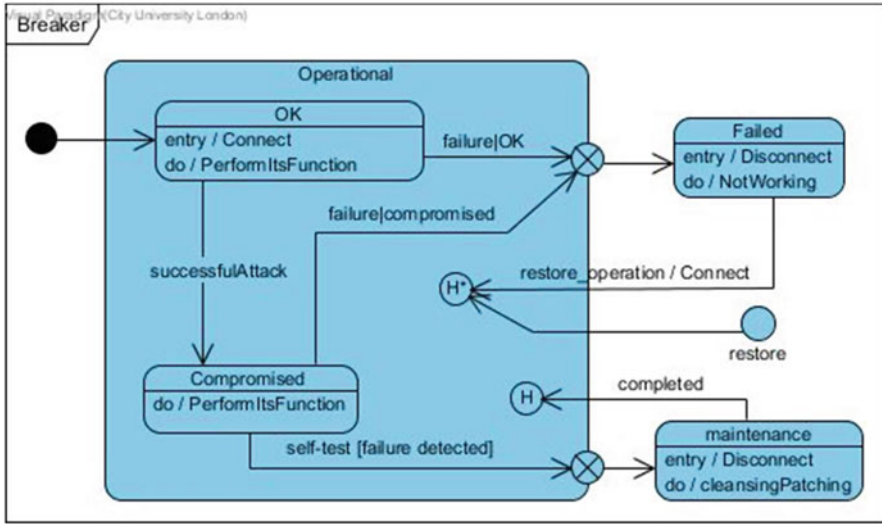


Fig. 4 A UML state machine diagram of a single channel protection device

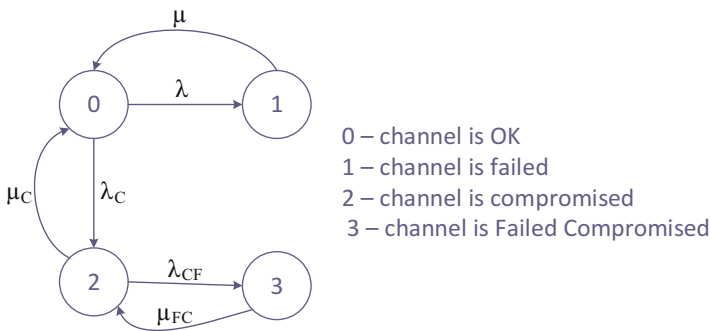


Fig. 5 A Markov chain diagram illustrating the behavior of a single channel protection device

on the state the device was in prior to failure (“OK” or “Compromised”); (ii) the maintenance state is not explicitly shown.⁵ The maintenance occurs while the device is in one of the failed states.

The transition rates are as follows:

λ – rate of failure in non-compromised state.

μ – rate of repair after a failure.

⁵The maintenance activity in the UML diagram is one which affects *all* of the devices across the network, and one via which transitions from the “Compromised” to the “OK” state in the Markov chain are realized. This could be modelled as a “shared activity”, but would require an adequate modelling support (e.g. available in Mobius SAN v2.5).

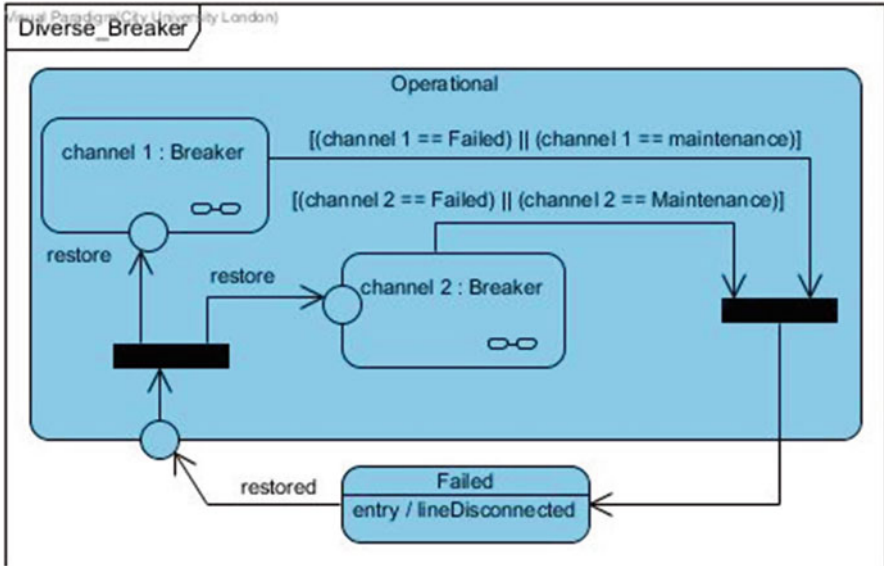


Fig. 6 UML state machine diagram depicting the behaviour of a 2-channel protection device

λ_C – rate of attack. This parameter is *related* to a system-wide rate of attacks (discussed later). During an attack, depending on an attacker’s preferences, a particular protection device is selected (e.g. no preferences, preferences for larger assets, etc.). As a result of such a selection, the rate of attack of a specific protection device is modelled as a fraction of the rate of attacks on the system.

μ_C – rate of inspection (i.e. of returning to OK state from a compromised state). This is a system-wide parameter. In our studies we ignore the time taken to inspect different devices and restore their operation after a compromise. We model the inspections by defining the distribution of times between successive inspections (e.g. exponential distribution with mean – a day, a week, etc.). The inspection is assumed to restore, *simultaneously*, the normal operation of all compromised devices.

λ_{CF} – rate of failure in compromised state.

μ_{FC} – rate of repair in Failed-Compromised state.

The behavior of a 2-channel protection device is shown in Fig. 6.

The 2-channel version implements a 1-out-of-2 architecture. That is, the 2-channel device fails only if both channels have failed. As long as at least one of the channels is operational (in either “OK” or “Compromised” state), the 2-channel breaker itself is also assumed to work correctly. The diagram uses the advanced features of UML 2.X to model the behaviour of each of the channels, including their possible failure, repair and maintenance. A channel can be restored on its own (triggered by the event “restore_operation”) or by a repair of the 2-channel system (triggered by “restored” event). In the former case the respective

channel is returned to the operational state it was in prior to this channel failure captured by “deep history” (H^*) pseudo-state.⁶ The latter case leads to forking a signal “restore” to both channels, which in turn returns each of the channels to the respective deep history pseudo-state. The “Breaker” state machine further models the device maintenance and eliminates the effects of a malicious compromise of the device via either the possibility of “cleansing” [6] – e.g. restoring the device to a known clean software configuration – or by patching it. In the model we assume that maintenance is always successful, hence it returns the state machine to OK state (modelled as “shallow history”, H , in the diagram, Fig. 4).

When a breaker fails, the corresponding component (line, generator, etc.) becomes *disconnected*. In either case, the failed protection device would respond to commands from the control centre (to connect or disconnect the respective line).

An external event, “successfulAttack”, triggers the transition “OK” \rightarrow “Compromised”. A return to the OK state requires maintenance.

In a “Compromised” state the protection device (or a channel, in the case of a two channel device) continues to operate, but it may fail in circumstances in which the device in an “OK” state would not, i.e. the *failure-rate* in a “Compromised” state is higher than it would be in an “OK” state. The rationale for this modelling choice is the desire to capture the effect of *advanced persistent threats* (APT), e.g. Stuxnet [15], under which the affected devices may continue to operate for some time before a failure occurs.⁷ This model of how cyber-attacks affect device (channel) behaviour in a compromised state is a special case of the model developed in [16], where the interested reader may find further detail.

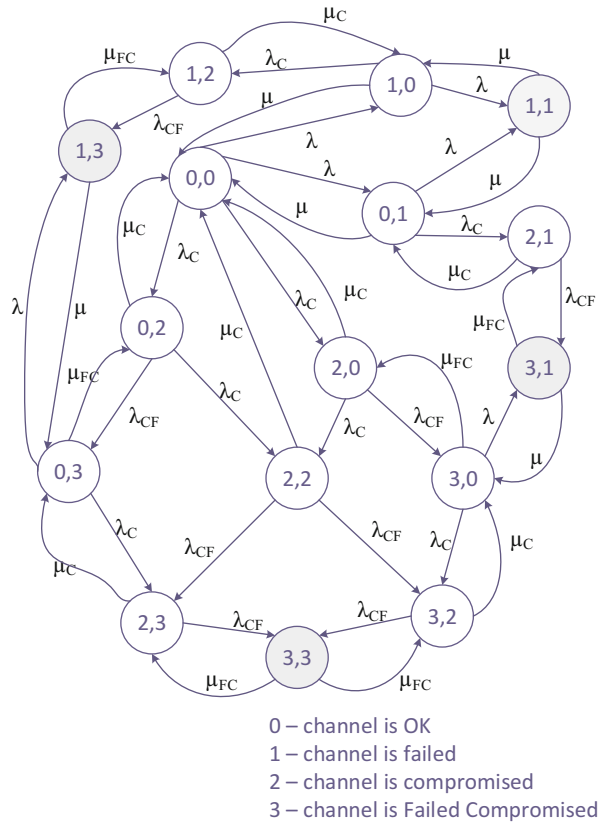
Figure 7 depicts a Markov chain which models the behaviour of the 2-channel protection device assuming exponentially distributed sojourn-times for all transitions. The state space of the chain is a Cartesian product of the channels’ state-spaces (defined in Fig. 5). The transitions correspond to the rates defined for the single channel device. The transition from state “2,2” to “0,0” captures the fact that the inspection of all devices is assumed an *atomic operation*, hence the states of both channels, when found “Compromised”, are changed simultaneously.

The shaded states, “1,1”, “1,3”, “3,1” and “3,3” are the states when the 2-channel device itself fails – both channels are in either “Failed” or “Compromised-Failed” state. According to our assumption that the device is a 1-out-of-2 protection device, when the chain is in one of these shaded states, the 2-channel protected device (a line, a generator or a load) is *disconnected* from the power network, which in

⁶The terms “deep history” and “shallow history” are part of the UML state-machine jargon. These refer to pseudo-states which are used with composite states (i.e. states which consist of two or more sub-states). The pseudo state “deep history” is used to signify that when a state machine enters a composite state, it will in fact enter the sub-state of this composite state in which the state machine was prior to leaving the composite state for the last time. The “shallow history” pseudo-state, instead, will always enter the same “initial” sub-state of the composite state.

⁷In fact, APT introduce subtle changes in the behaviour of the compromised devices (software), which are difficult to distinguish from normal operation, hence the compromised state may remain undetected for a long time.

Fig. 7 A Markov chain diagram illustrating the behavior of a 2-channel protection device. The labels attached to the states indicate the state of each of the channels, e.g. “2,0” means that channel 1 is in a “Compromised” state (“2”) and the second channel is in an “OK” state (“0”)



turn triggers a redistribution of the power in the power network, i.e. the power-flow calculation will be computed, which in turn may lead to more devices being disconnected if lines get overloaded.

Clearly, while in an operational state – “OK” or “Compromised” – the protection device works according to its specification: it either keeps the respective protected device connected to the power network or, when a power overload threshold is exceeded, the protection device will enact a powerline trip. This behaviour is not visible in the diagrams. Note that while the UML state-machine diagram captures normal operational behaviour – the device keeps the protected asset either connected to, or disconnected from, the power network – capturing these details with the Markov chain would be problematic. This is because transitions between operational sub-states (connected/disconnected) are triggered by changes of the entire power system – changes which are external to the protection device and typically follow deterministically after a change of the power network topology. External stimuli can be modelled with a UML state-machine while this is problematic for Markov chains.

We compare the effect on a system model's behaviour using two different models of a compromised breaker:

- (i) as soon as the line breaker is compromised its tripping threshold is set to a value which is 10% above the load to/from/through the protected asset (line, load or generator) at the time of the compromise. This tripping threshold can be significantly lower than the "correct" threshold, linked to the capacity of the respective asset, e.g. a line. We used this model in [4], following reports in the literature that similar attacks have indeed been observed [17]. The device failure may have no immediate consequence: it is only manifested once the state of the power network changes (e.g. as a result of accidental failure of a power component), resulting in the redistribution of electrical power flowing across the power network. If the flow through the protected asset then exceeds the incorrect tripping value set by a successful attack, the associated breaker will disconnect the asset;
- (ii) significantly increasing the rate of failure of a channel in a compromised state. An example would be an increase from a rate of failure *once in 10 years* in the non-compromised state to a rate of failure *once a day* in a compromised state (i.e. over *3 orders of magnitude*). Once the breaker fails (with a high failure-rate), under this model, it disconnects the respective protected element (a line, a generator or a load).

Clearly, the two models possess quite different levels of abstraction. The first one *deterministically* defines the breaker failure behaviour and requires detailed knowledge about the actions taken by an adversary. Such knowledge is only available for *known attacks*, for which extensive forensic analysis has been undertaken and their consequences established with certainty. Such knowledge, however, is not available for attacks which have *not* been seen or studied, e.g. those that use 0-day vulnerabilities. The first model, therefore, cannot be used for analysis in the face of *unknown attacks*. The second, more abstract model of a compromised breaker, instead, merely hypothesizes that a breaker compromise leads to a higher failure intensity, without defining specifics beyond the failure mode (that should the breaker fail it disconnects the protected component). Importantly, a lack of specific knowledge about *unknown attacks* is not, by itself, an impediment for using the abstract model to study the effects of these attacks.

Via suitable parameterization of the abstract model, can one reproduce system behaviour that is *close* to the behaviour arising when using the specific model of the compromised breaker instead? If it turns out that this is indeed possible, then there is an argument for using the abstract model in risk-assessment that includes unknown cyber-attacks. Varying the parameters of the abstract model might allow one to explore a spectrum of possible losses, without a detailed knowledge about how (unknown/future) cyber-attacks may compromise the respective devices.

3.3 Modelling Cyber-Attacks

Now we briefly describe the adversary model, which captures the behaviour of the attacker. This model is derived from [4] and is extended to capture the knowledge that an adversary may have about the deployed architecture of the breaker, e.g. whether defence-in-depth in the form of replicated breakers is deployed.

For the system under study we assumed that each substation has a dedicated firewall (indicated by the “brick wall” in Fig. 2), which isolates the sub-station from the rest of the world. We also assume that an intrusion detection/prevention system (IDS/IPS) monitors traffic in the sub-station’s LAN. When an IDS/IPS detects illegitimate traffic, it blocks an adversary from accessing those assets located at the substation.

Our study is limited to the effect of a *single type of attack* on the modelled system: a cyber-attack via the firewall of a sub-station. The model is shown in Fig. 8 using the *Stochastic Activity Networks* (SAN) formalism.

This model assumes that the adversary is periodically idle (represented by the SAN-place labelled “Idle”). With some regularity, defined by the *activity* *Attack_interval*, the adversary launches a cyber-attack on the system by trying to penetrate the Firewall (modelled in Fig. 8 by the activity *Firewall_attack*) of *one* of the 32 sub-stations defined in the NORDIC-32 model.

The selection of a substation to attack⁸ is driven by either a *uniform distribution*, defined over the 32 sub-stations (“Indiscriminate attacker profile”) or by a *non-uniform distribution* defined in a way to capture the *preferences* of the adversary, which are discussed elsewhere [18]. In this chapter, we limit the study to an indiscriminate adversary. Under the current model we also assume that the firewalls of all sub-stations are equally easy/difficult to penetrate. This model shows the steps that follow the adversary’s initial selection of a sub-station to attack:

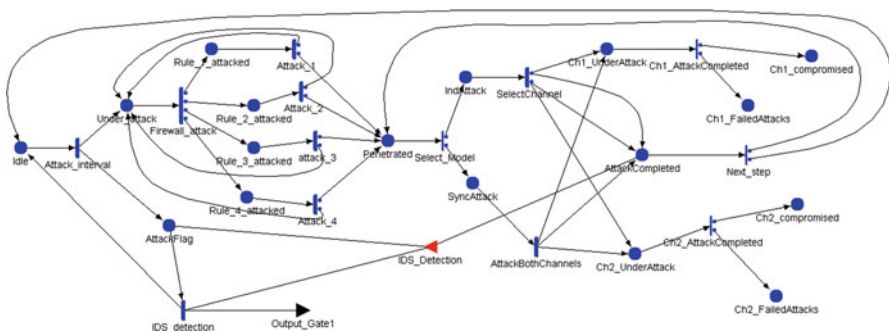


Fig. 8 Model of adversary applied to NORDIC-32

⁸Figure 8 does not show how the adversary chooses a sub-station.

- The adversary may target each of the firewall *configuration rules*. The decision of which rule to attack is modelled by the *activity* Firewall_attack. In Fig. 8 we assume that there are 4 rules to choose between, which is just an example. The model assumes that the rules are equally likely to be chosen by an attacker – the probabilities associated with the outputs of the Firewall_attack activity are all set to 0.25.
- Once a rule is selected (modelled by the places Rule_1 – Rule_4), the adversary spends time trying to break the selected rule. This is modelled by *activities* Attack_1 – Attack_4, respectively. Her efforts may be successful or unsuccessful. In the case of a failed attempt, the adversary returns to the state Under_attack and may try another rule.
- However, in the case of a successful penetration through the firewall, the state “Penetrated” is reached, in which case the adversary now has two further options for proceeding⁹:
 - compromise the protection device of a *single line*,
 - compromise the protection devices of *all lines* in the particular substation.

If the adversary succeeds, she leaves the substation. This choice is modelled by the *instantaneous activity* Next_step, which returns the adversary to the state “Idle”.

- When the protection device (breaker) is replicated, the adversary is presented with two further choices when in a “Penetrated” state (modelled by the “Select_Model” instantaneous activity):
 - *Independent attacks*: one of the channels of the breaker is selected at random and attacked. If the attack is successful, this channel enters the “ChX_compromised” state; the state of the second channel remains unaffected by the attack. This models the behaviour of an adversary *unaware* of the particular form of defence-in-depth (diverse replication of the breaker) she is facing. Note, with this adversary model, the two channels of the breaker may still eventually become simultaneously compromised, e.g. as a result of 2 separate attacks on the same substation, each attacking different channels of the same breaker;
 - *Synchronised attacks*: here, the adversary tries to compromise both channels of the breaker in the same attack, using suitably devised attacks for each of the channels. So each channel may be compromised as a result of a single attack. This adversary model captures an adversary with detailed knowledge of the deployed defence-in-depth. In this case the likelihood of compromising both channels of a protection device is clearly significantly higher than in the previous model with independent channel attacks.

⁹The actions that an adversary can take are not modeled in detail in Fig. 8. The specific logic of successful attacks – either changing the tripping threshold immediately or increasing the failure rate in the future, however, is implemented in the solver (simulator) of NORDIC-32.

- IDS/IPS is modelled by the *activity* `IDS_detection`, which is enabled if the model state is “Penetrated”. This activity *competes* with the activities for the adversary selecting and attacking the breaker channels. The adversary may be detected before she completes the attacks – as soon as the *activity* `IDS_detection` fires, the attack is aborted and the adversary is returned to “Idle”.

Finally, a channel of a protection device attacked multiple times may end up being *compromised multiple times* [16]. In this study, however, we ignore the implications of this possibility, assuming that the cumulative effect of multiple compromises of the same channel is no worse than the effect of a single successful attack; this is, admittedly, a simplifying assumption.

4 Results

First we compare the behaviour of the system model with two different adversary models: i) the adversary model described in [4]. Under this model, if the adversary succeeds in getting unauthorised access to the protection devices of a substation, she changes the tripping threshold from being set at a value slightly over the capacity of the protected device (line, generator and load) to a value that is merely 10% above the current flow through the protected device at the time of the successful attack; ii) the adversary model [16] discussed above. Under this model the rate of failure of a compromised protection device is set to a value of 10^3 greater than the rate of failure of the uncompromised device (i.e. before the successful attack on it).

In both cases a measure of interest is the expected value of the supplied power, as a fraction of the maximum power that can be supplied in the model, 10,940 MW. This measure is calculated via Monte-Carlo simulation. The NORDIC-32 system is simulated as running for 10 years of operation, repeated 300 times. The 300 simulation runs are a sample, allowing us to compute a sample estimate of the expected supplied power, as well as confidence intervals for this statistic at stated confidence levels.

In all simulated cases, in addition to those measures which seek to prevent an adversary from accessing assets of a substation, a periodic activity of “proactive recovery” (or cleansing) of the protection devices is in place. This activity restores the successful operation of protection devices by eliminating any effects of compromises that have taken place. When an adversary alters tripping thresholds, cleansing restores tripping thresholds to their nominal values. For the abstract model of a compromise, cleansing restores the failure-rates to the values assumed for non-

compromised states of the protection device. The cleansing procedure is assumed “perfect”, i.e. its outcome is always a success.¹⁰ Later in the chapter we discuss the implications of relaxing this assumption.

4.1 High Fidelity Vs. Abstract Adversary Models

In this section we summarise the results from the studies with the two adversary models. These are detailed in Table 1. The labels attached to the columns are as follows:

- μ represents the expected supplied power as a fraction of the nominal supplied power. The expectation is calculated over a number of simulation runs, N , typically 300. The average supplied power, P , is a random variable. For each simulation run, i , P takes some value p_i . We define μ as the expected value of P , and it is computed as: $\mu \equiv E [P] = \frac{\sum_{i=1}^N P_i}{N}$. Values of μ close to 1 (100%) represent cases with small average losses, while large deviations of μ from 100% indicate more significant losses, e.g. those due to cyber-attacks.
- σ is the standard deviation of P .

Table 1 Lost power due to attacks tampering with the tripping threshold of a protection device

	Case	μ	σ	LB	UB	p-value
Attacks change protection threshold	Daily attacks, no inspections	0.0319	0.0171	0.03	0.0338	<0.0005
	Weekly attacks, no inspections	0.2180	0.1143	0.205	0.2309	<0.0005
	Monthly attacks, no inspections	0.7185	0.2101	0.695	0.7423	<0.0005
	Yearly attacks, no inspections	0.9681	0.0552	0.962	0.9744	<0.0005
	Weekly attacks, inspect daily	0.9832	0.0015	0.983	0.9833	0.74823
	Weekly attacks, inspect weekly	0.9800	0.0028	0.980	0.9803	0.00070
	Weekly attacks, monthly inspections	0.9692	0.0089	0.968	0.9702	<0.0005
	Weekly attacks, yearly inspections	0.7653	0.1243	0.751	0.7794	<0.0005
	Attacks reduce reliability	Weekly attacks, no inspections	0.7913	0.0106	0.790	0.7925
Weekly attacks, monthly inspections		0.9772	0.0021	0.977	0.9774	0.90484
Weekly attacks, yearly inspections		0.9226	0.0185	0.920	0.9249	<0.0005
No attacks	Baseline	0.9845	0.0012	0.984	0.9846	0.53410

¹⁰Clearly, this is a simplifying assumption, which may not hold true in practice: the cleansing procedure itself may be fallible or it may be unavailable due to an insufficient number of personnel or insufficient amount of resource required for its enactment.

- LB and UB are the lower and upper bounds, respectively, of the 95% confidence interval for μ computed under the assumption that P is normally distributed.
- p-values are computed for the Anderson-Darling statistic, in a test for statistical normality. A value of the test statistic is computed for each sample of simulation runs (typically a sample-size of 300), and the associated p-value for the sample is the probability of observing a value for the test statistic that is no less extreme than the value computed from the sample, assuming the sample was indeed drawn from a normal distribution. This p-value should be compared with the required significance level, typically 0.05, to pass a judgement about normality – as values smaller than the significance level suggest that the hypothesis about normality should be rejected.

The top part of the table summarises the observations when the adversary model follows a specific cyber-threat closely – changing protection thresholds of the protection devices. Successful attacks of this kind have no immediate visible consequence and may manifest themselves only if/when the topology of the power network changes and the flow of power alters in such a way as to exceed the thresholds of some compromised devices and, thereby, trigger line trips. The problem may escalate over time, and unless the tripping thresholds are restored to their proper values the losses will be very significant, as the top 3 rows in the table indicate. Such large losses are clearly intolerable, and the problem with thresholds is likely to be identified and fixed. For this reason, although the studies point to a potentially serious type of attack, the fix is relatively simple.

Looking at those rows of the table which summarise the effect of inspections, one sees that the frequency of inspections affects losses, which is not surprising. Monthly inspections leave the losses within 2% of the baseline – 0.9692 vs 0.9841 for the average supplied power.

Let us now compare the model results with the two models of attacks – using a detailed model of stealth attacks vs the more abstract model of the effect attacks would have on compromised protection devices. The two highlighted rows of the table show losses calculated with the two models. It is striking how close the average losses are: 0.9692 for the stealth model vs. 0.9772 for the abstract model of the compromised protector. Although the difference between these averages is statistically significant¹¹ the absolute difference is negligible – less than 1%! This observation suggests that despite conceptual differences between these models of how attacks compromise protection devices, the average losses from stealth attacks for the particular case (of weekly attacks and monthly inspections) for this particular system, NORDIC-32, can be *estimated quite accurately* using a model which operates at a much higher level of abstraction. And more importantly, the abstract model does not rely on detailed knowledge of the mechanisms of how the

¹¹ We do not present the results from testing statistically the hypothesis that the means of the two samples are the same, but did conduct this test and the null hypothesis was strongly rejected.

stealth attacks might alter the behaviour of the protection devices, which makes the abstract models potentially very attractive for assessing the risk from future, unknown attacks.

Perhaps it is noteworthy that these results were “*easily*” obtained from an initial *informal exploration* of the abstract model’s parameter space. We ran a short campaign with an order of magnitude increase of the failure rate as a result of a compromise. The effects on the system model were negligible. We then tried an increase of 3 orders of magnitude and this choice of parameterisation for the abstract model produced the agreement between the models we report here.

In practice, the parameterisation of the abstract model is likely to be done more systematically. Here, we list a number of options worth considering:

- one may carry out *systematic sensitivity analysis* exploring how failure-rate¹² increases affect system behaviour. In case specific models of past cyber-attacks are available, one could try to identify a range of parameters for the abstract model, for which the system model behaves comparably to how it behaves when the more detailed models of compromise are used, e.g. repeat, for a whole slew of known attacks, a study similar to the one we reported above. Selecting the abstract model’s parameter values from within this range might give some indication about the system’s preparedness against both known (which is usually where security assessment stops!) and unknown cyber-threats which happen to have consequences that are captured accurately enough by the abstract model parameterised from the range identified in the sensitivity study.
- Clearly with the abstract model of consequences, the failure-rate may increase to infinity, which would result in an instantaneous failure of the compromised device. Instead of using the failure rate increase (i.e. a parameter, relative to the rate of failure of the non-compromised device), one may parameterise the abstract model using an *absolute failure-rate*. With this, a sensitivity analysis may still be employed to determine a useful range of values: from instantaneous failure to a rate which corresponds to mean-time-to-failures of a few units of meaningful time, e.g. seconds, minutes or hours, depending on the specific context.

¹²Clearly, by referring to the rate of failure, we implicitly envisage an exponential distribution of the time to failure, which is often used as it reduces the problem of parameterisation to a single value. Should there be a reason ruling out the use of exponential distribution, the abstract model parameterisation will become more complex. It will involve a selection of a suitable family of probability distributions and applying sensitivity analysis to their respective parameters.

4.2 *Quantification of Defence-In-Depth Using the Abstract Model*

In this section we look at the effect of applying defence-in-depth (DiD) in the form of 2- channel protection devices deployed at certain points across the network, instead of 1-channel protection devices. The options that we considered are: applying DiD to devices protecting lines only, generators only, loads only or all power elements (lines, and generators, and loads).¹³

In all of the cases with 2-channel protection devices we study the behaviour of the system model subjected to different attacks on protection devices:

- *Independent attacks*: each time an adversary succeeds in gaining access to a 2-channel protection device, she compromises *only one of the channels*, selected at random. Under this mode of attack, compromising both channels is still possible, but would require at least two separate successful attacks on each of the channels of the same device, respectively.
- *Synchronised attacks*: every successful attack on a protection device results in both channels being compromised (i.e. simultaneously by the same attack).

As stated above, we ignore the effects of the second, third, etc., successful attacks on the same channel, a simplifying assumption made for convenience (to reduce the number of modelling parameters). Under rather broad conditions, reducing the periods between proactive recoveries will reduce the probability of multiple compromises of the same protection channel, which provides some justification for the adopted simplification. Clearly, a sufficiently frequent “proactive recovery” reduces the probability of a protective channel being compromised more than once to a negligibly small number, justifying ignoring the effects of multiple attacks on the same protection device.

The results from our studies are reported in Table 2 and grouped according to the *mode of attack* – independent or synchronised, whether inspections (i.e. cleansing) are applied or not and, if applied, the rate of the inspections. At the bottom of the table, the simulation results are presented for a system with single channel protection devices. This last case is included to demonstrate some of the benefits from DiD.

Comparing the three rows labelled “Baseline” clearly indicates that the cases are statistically indistinguishable from the point of view of the selected measure of interest (supplied power): the collected measures are practically identical. Statistical tests of whether the samples from the simulation runs, collected for all 3 cases, come from the *same distribution*, provided us with no evidence to suggest that the hypothesis should be rejected. This observation is somewhat surprising, as it suggests that using replicated protection devices brings *no benefits* for the modelled

¹³Clearly, limiting the total number of 2-channel protection devices, and trying to identify the optimal places to deploy these resources in the system, is yet another example of a worthwhile study.

Table 2 Defense-in-depth: Independent vs. synchronized attacks on protection devices

System model		μ	σ	LB	UB	p-value		
Independent attacks	Baseline (attacks disabled)		0.9845	0.0012	0.984	0.985	0.534	
	No inspections	Weekly attacks (all)	0.9443	0.0079	0.943	0.945	0.481	
		Weekly attacks (generators)	0.9529	0.0025	0.953	0.953	<0.005	
		Weekly attacks (lines)	0.9577	0.0046	0.957	0.958	0.3977	
		Weekly attacks (loads)	0.9629	0.0013	0.963	0.963	0.4911	
		Monthly inspections	Weekly attacks (all)	0.9843	0.0012	0.984	0.984	0.145
		Weekly attacks (generators)	0.9837	0.0013	0.984	0.984	<0.005	
		Weekly attacks (lines)	0.9843	0.0011	0.984	0.984	0.4103	
		Weekly attacks (loads)	0.9838	0.0012	0.984	0.984	0.008	
		Yearly inspections	Weekly attacks (all)	0.9801	0.0036	0.98	0.980	<0.005
	Weekly attacks (generators)		0.9702	0.0043	0.97	0.971	0.117	
	Weekly attacks (lines)		0.9816	0.0023	0.981	0.982	<0.005	
	Weekly attacks (loads)		0.9752	0.0027	0.975	0.975	0.003	
	Synchronised attacks	Baseline (attacks disabled)		0.9845	0.0012	0.984	0.985	0.416
		No inspections	Weekly attacks (all)	0.8930	0.0057	0.892	0.894	0.418

(continued)

Table 2 (continued)

System model		μ	σ	LB	UB	p-value	
	Weekly attacks (generators)	Weekly attacks (generators)	0.9505	0.0016	0.950	0.951	0.187
		Weekly attacks (lines)	0.9264	0.0037	0.926	0.927	0.884
		Weekly attacks (loads)	0.9609	0.0011	0.961	0.961	0.462
	Monthly inspections	Weekly attacks (all)	0.9810	0.0014	0.981	0.981	0.518
		Weekly attacks (generators)	0.9774	0.0016	0.977	0.978	0.278
		Weekly attacks (lines)	0.9825	0.0013	0.982	0.983	0.718
		Weekly attacks (loads)	0.9798	0.0012	0.980	0.98	0.056
	Yearly inspections	Weekly attacks (all)	0.9560	0.0089	0.955	0.957	<0.005
		Weekly attacks (generators)	0.9588	0.0036	0.958	0.959	0.980
		Weekly attacks (lines)	0.9667	0.0063	0.966	0.967	<0.005
		Weekly attacks (loads)	0.9672	0.0018	0.967	0.967	0.929
	1-channel protection device	(all) weekly attacks, no inspections	0.7912	0.0106	0.79	0.792	0.032
		(all) weekly attacks, monthly inspections	0.9772	0.0021	0.977	0.977	0.905
(all) weekly attacks, yearly inspections		0.9226	0.0185	0.920	0.925	<0.005	
Baseline (1-channel protection device)		0.9845	0.0012	0.984	0.985	0.302	

system, provided the system operates in a *trusted environment* without attacks. The reason might be that the rate of failure of the protection devices is very low (MTTF 10 years), which makes redundancy unlikely to improve a device's reliability in the face of accidental failure.

The rows at the bottom of Table 2 (labelled 1-channel device) provide measures from the attack and inspection rates used to study DiD: weekly attacks and no inspections, monthly and yearly inspections. A comparison of 1-channel and of 2-channel protection devices indicate clear benefits from employing replication in a trusted environment. The benefits are more clearly pronounced for independent attacks.

Now let us compare the model behaviour with, and without, DiD, and under different attack modes: independent and synchronised attacks.

- No inspections.
 - Without inspections, the losses under independent attacks are clearly smaller than the losses under synchronised attacks: the expected value of supplied power under independent attacks is closer to the values recorded for the Baseline studies than the losses from synchronised attacks.
 - Stratification – attacks are applied to all protection devices vs. to generators only, lines only and loads only – provides additional insight as to where DiD would bring the most serious benefits. Under the independent attacks model, losses from attacks on generators are greater than the losses from attacks on the lines or on loads. Under synchronised attacks, however, the pattern is different. With no inspections the largest losses from synchronised attacks are recorded for attacks on the lines, while the losses from attacks on generators are lower than from attacks on both lines and loads.
- Inspections (either monthly or yearly). Adding inspections changes the ordering between the cases quite subtly.
 - For independent attacks, even yearly inspections make the system comparable to the Baseline case: the additional power lost due to weekly cyber-attacks is only a small fraction of a percent. Clearly, the combination of replication in protection, together with the favourable attack regime (one channel at a time), is sufficient for the effects of cyber-attacks to be *almost entirely compensated*; the additional losses are very small. Increasing the rate of inspections (monthly) reduces the additional losses due to cyber-attacks even further, which is not surprising.
 - For synchronised attacks the fact that the two channels of a protection device can be compromised by the same attack, leads to device failure shortly (on average 7.5 h later) after a successful attack. A device failure, in turn, leads to disconnecting the respective protected component (a generator, a line or a load) from the power network, i.e. the topology of the power network changes, and some power losses become inevitable. Yearly inspections are simply not frequent enough to mitigate the additional losses due to cyber-attacks: with yearly inspections the losses due cyber-attacks are almost 3 times greater than they are due to accidental failures (the baseline case). Our results suggest that

monthly inspections can mitigate – to a large extent – the additional losses: the model behaviour with monthly inspections is very close to the Baseline case, especially for the cases when power-line protections are under attack. Intuitively, this last observation is not surprising: disconnecting some lines may be of no immediate consequence. Whether disconnecting a line will lead to losses or not depends on the topology of the power network before and after disconnecting the line. Our study also suggests that the monthly inspections are less effective in mitigating losses from attacks on protection devices attached to generators and loads. Although intriguing, this observation is not surprising either: disconnecting a load in the power network leads to an immediate loss of power. The effect of disconnecting a generator is less obvious: in some cases the effect may be nil, e.g. if the operational generators have spare capacity sufficient to pick up the required power and the topology of the network is such that it does not get overloaded. If a large generator is disconnected,¹⁴ however, a power loss is imminent and substantial.

5 Discussion

Our studies demonstrate the quantitative analysis of cyber-risks in a complex industrial system. Contrary to a commonly adopted approach to cyber-risk assessment (e.g. [19], relying on “high”, “moderate”, or “low” qualitative indicators of impact), we demonstrate that the impact of cyber-attacks can be meaningfully established using a model of the *particular cyber-physical system*. While we share the view that establishing the likelihood of various cyber-attacks is difficult and, perhaps, unknowable,¹⁵ the quantitative method of cyber-risk assessment we put forward here seems useful. Dismissing quantitative methods because of a lack of credible methods to capture likelihoods seems to miss the point. Yes, even if, somehow, the “true” likelihoods of attacks can be captured today, these are likely to become hopelessly inaccurate when the landscape of cyber-threats changes tomorrow. However, instead of giving up on quantitative risk-assessment because of this difficulty, one could opt for performing sensitivity analysis over a range of plausible likelihoods. Using such an approach could establish useful bounds for risk indices of interest (e.g. the lost power in our studies). This is much better than using questionable indices with values {high, moderate, low} calculated on a scale devoid of mathematical rigour, and that typically ignore the specific application context!

Now, arguably, there is a fundamental issue with security assessment activities that are solely based on establishing whether “best practice and engineering principles” have been followed. While there is no doubt that such assessment approaches are sensible, they do fall short in answering the question of whether the

¹⁴One of the generators in NORDIC-32 provides more than 40% of the power in the network. The smallest generator – provides less than 10% of the total power.

¹⁵This is a “known unknown”.

system is “secure enough”. Clearly, while undertaking an assessment (certification) gives some confidence that the system is prepared against anticipated (i.e. known) attacks, such confidence can be misleading, especially if the system scores very well in the assessment (certification). The problem is the assessment provides no indication of how good the system defences are against *unknown* cyber-threats (e.g. those that exploit 0-day vulnerabilities).

One approach to tackling this problem was developed recently in [16], which we have now attempted to validate here. By using an abstract adversary model consistent with [16], we reproduce the expected power loss experienced by the NORDIC-32 power system subjected to sophisticated stealth attacks. Here, with the power system undergoing monthly maintenance inspections and being subjected to weekly attacks, each successful attack resulted in a modified tripping threshold for some protection device. The corresponding abstract adversary model does not explicitly represent these threshold changes; instead, the consequence of a successful attack is represented as an increase in the failure rate of the affected device. And yet, the expected losses, 0.9692 and 0.9772, under these very different alternative ways of capturing the effects of successful attacks, are in close agreement. Our studies highlight the potential for the behaviour of a CPS, subjected to a previously unknown sophisticated cyber-attack, to be suitably mirrored by subjecting the CPS to attacks from a properly parameterised abstract adversary model. Fully demonstrating such a substitute of “the specific” with “the abstract” will take more than our simulation studies, but we believe the work we report here is important as it indicates a useful way forward in addressing unknown cyber-attacks. Finding out how the system might be affected by unknown attacks may prompt system operators to look for additional controls to bring risk down to an acceptable level.

The final set of results – quantifying the effect of DiD – is also intriguing. With these we confirm the observations made in [16], but with a much larger CPS – that a model of an adversary attacking replicated assets (in this case protection devices) is significantly affected by the adversary’s knowledge of the architecture of the assets. The improvements DiD bring against *independent attacks* (i.e. when a single channel of a replicated asset is attacked) are more significant than the improvements against synchronised attacks. The magnitude of the difference depends on how replication is complemented by inspections – measures to cleanse software from the effects of cyber-attacks.

The main message of the study, however, is in demonstrating the feasibility of deploying DiD *rationally*. If an operator has identified a number of plausible and affordable alternatives, say A, B, C, etc. to deploying DiD, then she doesn’t need to count on “gut feelings” to choose amongst these. No; instead, she can run model-based studies with each of the available alternatives and compare the resulting improvements. Such studies, however computationally demanding, are a small price to pay in comparison with investing in a sub-optimal alternative that gives little to no improvement. The feasibility of the approach is demonstrated in this work: we identified a number of alternative deployments of DiD – all equipment protected, protection for only generators, or lines, or loads – and report on the benefits each of these candidate DiD-deployments bring.

6 Related Research

In addition to the references given earlier, we would like to outline a number of related sources.

There have been studies applying different modelling techniques to *known* attacks. A couple of examples are [20, 21]. The first reference applies a probabilistic technique to define a model of Stuxnet, and demonstrates how model parameters can be assigned plausibly. The second example, instead, uses a non-probabilistic formalism. These authors claim that documenting the particular malware is, itself, an important contribution. Neither of the two models, however, is used by the respective authors for an analysis of open research problems. Our focus is quite different here: instead of merely constructing a model of something that has been seen, we use a model as a tool to study practical problems such as the effectiveness of DiD in different, adverse environments.

Somewhat related to the work presented in this chapter is our own previous work on modelling the effect of cyber-attacks on the reliability of an embedded device with fault-tolerant software [22]. The style of modelling there and in this chapter are conceptually similar, but the scope of the analysis is quite different. The tools to implement the work are also quite different. In [22] we developed a detailed model of a specific device – to study a specific attack on the safe-state of the device – using the *stochastic activity networks* (SAN) formalism. In this chapter, however, we use complex hybrid models of power systems which combine both probabilistic (stochastic state machines) and deterministic (e.g. power-flow models) parts. A SAN is depicted in Fig. 8 merely as an aid in describing the adversary model.

The synchronized attacks that we studied in detail are *conceptually similar* to common mode/cause failure; a topic which has been studied extensively in the context of system/software safety and highly available computer systems.

There is also conceptual similarity in the proposed approach of replacing specific models of adversaries with more abstract counterparts and the popular approach to dependability analysis based on fault injection – trying to learn about true faults via injection of faults believed by the proponents of the methods to be representative. Despite the conceptual similarities – replacing “reality” with surrogates – there are significant conceptual differences. Many of the fault-injection based studies merely assume that injecting faults is a valid approach. In our work, we try to gain confidence in those model parameter values potentially related to unknown attacks by identifying those parameter values which make the abstract model suitably mimic the “real thing”.

Finally, we would like to acknowledge the ADVISE formalism, a part of the popular Mobius tool (<https://www.mobius.illinois.edu/>). The ADVISE formalism captures, probabilistically, the motivation of an adversary, the assets of a particular system and the rewards that successful attacks will bring to the successful adversary. ADVISE operates at a high level of abstraction, which may pose some difficulties in estimating risk indices requiring detailed causal mechanisms for their computation, such as the expected loss of power which we used in our studies. Modelling

synchronized attacks, which require detailed knowledge of defense-in-depth, with ADVISE is likely to pose additional difficulties too.

7 Conclusions and Future Research

This chapter provides a number of results concerning a quantitative assessment of cyber-risks in cyber-physical systems (CPS) – one which we proposed a few years ago. We use a complex model of the power-transmission system, NORDIC-32, extended with measurement, protection and control, all in line with the recent standard for interoperable sub-stations, IEC 61850.

We report on two important advances:

- experimental evidence that, via suitable parameterisation of an abstract model, the expected losses due to a specific attack can be established fairly accurately. This result is significant as it points to an intriguing prospect of quantifying risks from unknown cyber-attacks.
- demonstrating that our model-based approach can be used to support rational, evidence-based decisions, about how to maximise the benefits from investing in defence-in-depth (DiD). We studied the effectiveness of DiD – a combination of design diversity in protection devices and proactive recovery of the channels – as a defence against two types of attackers:
 - a naïve attacker, unaware of the nature of DiD, who would select only one of the channels of the protection device to compromise at any one time, and,
 - a knowledgeable attacker. One with detailed knowledge of the DiD they face, and able to launch attacks which defeat the DiD.

This work can be extended in a number of ways. The encouraging result, that there are easily identifiable circumstances under which an abstract adversary model can be used to accurately establish losses from a fully defined attacker, needs further scrutiny. In what ways can this be harnessed to give more insight into unknown attacks? In part, this will require exploring more specific models of attacks, studying how well the abstract model can mimic these, and using sensitivity analysis to establish ranges of the abstract model's parameters that result in plausible, but as-of-yet unseen, cyber-attacks.

It is unclear at this stage whether the abstract adversary model used in this work is universally applicable, i.e. whether accurate estimates of the loss can be achieved for any attack type – we suspect not. The modelling circumstances under which such parity can be accomplished, as well as the generality of the approach, requires further investigation. In order to shed more light on the problem, we plan to look at sophisticated attacks, e.g. compromising WAMS software (mentioned earlier) or other “special purpose software” (SPS). Such cyber-attacks could lead to system operators being presented with plausible, but nevertheless incorrect, data on the state of the CPS, causing these operators to take erroneous decisions in the control room.

It may well turn out that the effects of compromised SPS require a different family of abstract models. Constructing these with the same objective – getting accurate estimates of losses due to attacks on SPS – is an important direction for immediate future research.

Acknowledgement This work was supported by the UK EPSRC CEDRICS project, part of the UK Research Institute of Trustworthy Industrial Control Systems (RITICS), by the UK GCHQ and by the AQUAS project funded in part by the EU ECSEL – JU Programme (project ID 737475).

A.1 Appendices

A.1.1 Appendix A: Model of Power Line

```

1   {
2   "name": "Link",
3   "type": "state-machine",
4   "comment": "Represents physical lines between
               substations. ",
5   "properties": {
6     "from": {
7       "type": "Lookup",
8       "required": true,
9       "properties": {
10        "list": "machines",
11        "filter": "name === 'Substation'",
12        "value": "name"
13      }
14    },
15    "to": {
16      "type": "Lookup",
17      "required": true,
18      "properties": {
19        "list": "machines",
20        "filter": "name === 'Substation'",
21        "value": "name"
22      }
23    },
24    "kV": {
25      "type": "String",
26      "required": true
27    },
28    "x": {
29      "type": "Number",
30      "required": true
31    },
32    "max": {
33      "type": "Number",
34      "required": true
35    },

```

```

36     "overloaded": {
37         "type": "Boolean",
38         "required": true
39     },
40     "connected": {
41         "type": "Boolean",
42         "required": true
43     },
44     "failure": {
45         "type": "Activation",
46         "required": true
47     },
48     "recovery": {
49         "type": "Activation",
50         "required": true
51     },
52     "length": {
53         "type": "Number",
54         "required": true
55     }
56 },
57 "structure": {
58     "states": [
59         "ok",
60         "fail"
61     ],
62     "initial": "ok",
63     "transitions": {
64         "ok": {
65             "fail": [
66                 {
67                     "type": "property",
68                     "property": "failure"
69                 }
70             ]
71         },
72         "fail": {
73             "ok": [
74                 {
75                     "type": "property",
76                     "property": "recovery"
77                 }
78             ]
79         }
80     }
81 }

```

This code fragment provides the definition of a Power Line and includes the respective state machine and a set of properties defined for the line.

A.1.2 Appendix B: A Detailed Description of Attacks on a Breaker

```

1.  {
2.    "name": "Breaker Component",
3.    "type": "state-machine",
4.    "structure": {
5.      "states": [
6.        "ok",
7.        "fail",
8.        "compromised-ok",
9.        "compromised-fail"
10.     ],
11.     "initial": "ok",
12.     "transitions": {
13.       "ok": {
14.         "fail": [
15.           {
16.             "type": "probabilistic",
17.             "distribution": "exponential",
18.             "parameter": 0.1,
19.             "comment": "once in 10 years"
20.           }
21.         ]
22.       },
23.       "fail": {
24.         "ok": [
25.           {
26.             "type": "deterministic",
27.             "parameter": 0.00086,
28.             "comment": "7.5h"
29.           }
30.         ]
31.       },
32.       "compromised-ok": {
33.         "compromised-fail": [
34.           {
35.             "type": "probabilistic",
36.             "distribution": "exponential",
37.             "parameter": 365,
38.             "comment": "daily"
39.           }
40.         ]
41.       },
42.       "compromised-fail": {
43.         "compromised-ok": [
44.           {
45.             "type": "deterministic",
46.             "parameter": 0.00086,
47.             "comment": "7.5h"
48.           }
49.         ]
50.       }
51.     }
52.  }

```

The code fragment (in json notation) defines a state machine, which captures the adversary behaviour. The state machine definition starts in line 4, from which its structure is defined: i) the *states* (“ok”, “fail”, “compromised-ok” and “compromised-fail”), “ok” is defined as the initial state, and ii) the *transitions* between the states, which define the source and destination state for each of the transitions, together with a *distribution* of the transition duration: distribution type and the parameters, required by the respective distribution type. Most of the transitions in this example are assumed exponentially distributed: this distribution requires a single parameter. The recovery from a failure (with or without a compromise) is deterministic: a fixed duration of 7.5 h, a somewhat arbitrary figure. Apart from these two options – exponentially distributed and deterministic – a number of alternatives distributions for the transition durations are available to a modeller to choose from.

References

1. Eckhardt DE, Lee LD (1985) A theoretical basis for the analysis of multiversion software subject to coincident errors. *IEEE Trans Softw Eng* SE-11(12):1511–1517
2. Popov P, Littlewood B. (2004) The effect of testing on reliability of fault-tolerant software. In: *Dependable Systems and Networks (DSN’04)*. IEEE Computer Society Press, Florence
3. DHS, I.-C (2016) Recommended practice: improving industrial control system cybersecurity with defense-in-depth strategies, 58. Available from: https://ics-cert.us-cert.gov/sites/default/files/recommended_practices/NCCIC_ICS-CERT_Defense_in_Depth_2016_S508C.pdf
4. Netkachov O, Popov PT, Salako K (2016) Model-based evaluation of the resilience of critical infrastructures under cyber attacks. In: Ellinas G, Panayiotou C, Kyriakides E, Polycarpou M (eds) *Critical information infrastructures security (CRITIS 2014)*. Springer, Limasol, pp 231–243
5. Sousa P et al (2010) Highly available intrusion-tolerant services with proactive-reactive recovery. *IEEE Trans Parallel Distrib Syst* 21(4):452–465
6. Arsenault D, Sood A, Huang Y (2007) Secure, resilient computing clusters: self-cleansing intrusion tolerance with hardware enforced security (SCIT/HES). In: *2nd International conference on availability, reliability and security*. IEEE Computer Society Press, Los Alamitos, CA
7. Teixeira A et al (2011) A cyber security study of a SCADA energy management system: stealthy deception attacks on the state estimator*. *IFAC Proc* 44(1):11271–11277
8. Liu Y, Ning P, Reiter MK (2009) False data injection attacks against state estimation in electric power grids. In: *Proceedings of the 16th ACM conference on computer and communications security*. ACM, Chicago, Illinois, USA, pp 21–32
9. Christensen CM (1997) *The innovator’s dilemma: when new technologies cause great firms to fail*. Harvard Business School Press, Boston
10. Netkachova K et al (2015) Using structured assurance case approach to analyse security and reliability of critical infrastructures. In: *SAFECOMP 2015: ASSUREworkshop*. Springer, Delft, Netherlands, pp 345–354
11. Stubbe CM (1995) Long term dynamics, phase II. *CIGRE TF* 38.02.08
12. Peppas D (2008) *Development and analysis of Nordic32 power system model in powerfactory in school of electrical engineering, electric power systems*. Royal Institute of Technology, Stockholm, p 77
13. Bloomfield RE et al (2017) Preliminary interdependency analysis: an approach to support critical-infrastructure risk-assessment. *Reliab Eng Syst Saf* 167:198–217

14. Netkachov O (2018) HPS: high performance simulation engine of cyber-physical systems. Available from: <http://openaccess.city.ac.uk/19330/>
15. Falliere N, Murchu LO, Chien E (2011) W32.Stuxnet Dossier, 69. Available from: http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf
16. Popov P (2017) Models of reliability of fault-tolerant software under cyber-attacks. In: The 28th IEEE international symposium on software reliability engineering (ISSRE'2017). IEEE, Toulouse, p 228
17. Zetter K (2016) Countdown to zero day: Stuxnet and the launch of the world's first digital weapon. Broadway Books (A Division of Bantam Doubleday Dell Publishing Group Inc); Reprint edition (15 Sept. 2015) New York, p 448
18. Netkachov A, Popov P, Salako K (2014) Quantification of the impact of cyber attack in critical infrastructures. In: 1st International workshop on reliability and security aspects for critical infrastructure protection (ReSA4CI 2014). Springer, Florence (co-located with SAFECOMP 2014)
19. ISA (2017) ISA-62443-3-2, security for industrial automation and control systems: security risk assessment, system partitioning and security levels. International Association of Automation (ISA), p 38
20. Kriaa S, Bouissou M, Pietre-Cambacedes L (2012) Modeling the Stuxnet attack with BDMP: towards more formal risk assessments. In: Martinelli F et al (eds) 7th International conference on risks and security of internet and systems (CRiSIS). IEEE, Cork, p 8
21. Maynard P, McLaughlin K, Sezer S 2016 Modelling Duqu 2.0 malware using attack trees with sequential conjunction. In: 2nd International conference on information systems security and privacy. SciTePress, Rome
22. Popov PT (2015) Stochastic modeling of safety and security of the e-Motor, an ASIL-D device. In: Koornneef F, van Gulijk C (eds) 34th International conference on computer safety, reliability, and security (SAFECOMP 2015). Springer, Delft University of Technology, Delft, pp 385–399

A Model-Driven and Generative Approach to Holistic Security



Frederik Gossen, Tiziana Margaria, Johannes Neubauer,
and Bernhard Steffen

Abstract Functional and technical cyber-resilience gain increasing relevance for the health and integrity of connected and interoperating systems. In this chapter we demonstrate the power and flexibility of extreme model-driven design to provide holistic security to security-agnostic applications. Using C-IME, our integrated modelling environment for C/C++, we show how easily a modelled application can be enhanced with hardware security features fully automatically during code generation. We illustrate how to use this approach and design environment to make any modelled application ready to securely store its data in potentially insecure environments. The same approach can be used to secure communication over potentially insecure channels. In fact, our approach does not require any changes of the application model. Rather, our integrated modelling environment provides a dedicated modelling language for code generators which resorts to a Domain Specific Language for security. It is realized as a palette of security primitives whose implementation is based on underlying hardware security technology. The code generator injects security appropriately into the models of the applications under development. We illustrate the use of this security-injecting code generator on the case study of a to-do list management application. The code generator is generic and can be used to secure the file handling of any application modelled in the C-IME.

F. Gossen (✉) · T. Margaria
University of Limerick and Lero – The Irish Software Research Centre, Limerick, Ireland
e-mail: frederik.gossen@lero.ie; tiziana.margaria@lero.ie

J. Neubauer · B. Steffen
TU Dortmund University, Dortmund, Germany
e-mail: johannes.neubauer@tu-dortmund.de; bernhard.steffen@tu-dortmund.de

1 Introduction

In a connected world, security becomes increasingly important as cyber-attacks emerge quickly on the wealth of new services [9]. Cyber-resilience has been defined as “*the ability to continuously deliver the expected outcome despite adverse cyber-events.*” [2]. Functional and technical integrity play here a particularly important role, concerning the ability to deliver the expected processes and capabilities in the context of an IT system, including communication channels and networks. Cloud-based applications are one example of particularly vulnerable applications as they store their data online where it is ideally accessible from anywhere, but only to authorized users. However, there are numerous cases in which these environments have proven to be insecure. Third parties gained access to data stored in widely used cloud storage services [8]. One way to protect this data is to enhance security mechanisms of the environment, but this is only possible within one’s own scope of control. Popular cloud storage services such as Dropbox [7], Google Drive [14] or OneDrive [34] are prominent examples for uncontrolled environments. As we are seeing with increasing frequency also in embedded and cyberphysical systems, the temporary or permanent storage of data coming from sensors is a hot topic. It impacts for instance the trust relations underlying sensor and decision-making based proactive maintenance of remote machines: the sensor data is increasingly often moved elsewhere for processing and decision-making, often in some form of cloud and grid computing, in a trust space encompassing machine producer, machine operator, maintenance service provider, and one or more levels of customers. Depending on the nature of the data, security, privacy, confidentiality may all play a role. The risk structure, and as a consequence the resilience of the entire complex system, hinge on adequate, holistic and integrated protection mechanisms that go beyond the most frequently practiced but inadequate patchwork of individual technologies at the different layers of its subsystems.

Rather than relying on secure storage services, one can protect data by encryption before it is uploaded. However, this native encryption requires secure operations to be used by applications on every interaction with the data. For example, use secure file operations whenever interacting with the file system. From the developer perspective, security is thus a prime example of a cross-cutting concern: it requires all file operations across the entire application to be consistently implemented in a secure manner. This globality of scope makes it extremely difficult to add security features a posteriori, once critical parts of the application are already implemented, and it makes a case for handling security as an ab-initio design and quality dimension.

Although some security concerns are often dealt with in the early development process, many security requirements become known only much later, often even after deployment [6, 9]. Thus we need methods that can cope with this conflicting requirement: be holistic, but be able to deal comfortably with evolution and change, as along the long life of complex systems and applications it will be needed to security-enhance and security-upgrade applications which are already

fully functional. The same problem and need applies to secure communication. Our model-driven approach can be used to tackle communication in the very same way.

In this chapter we show how an application modelled in C-IME [15], our integrated modelling environment for C/C++, can be enhanced with hardware-based security features without even touching its models. This ease demonstrates the power and flexibility of extreme Model-Driven Design (XMDD) [28, 30], the design and development approach embodied by all our integrated design environments. C-IME itself is the product variant specialized for C/C++ support in a product line of integrated modelling environments¹ (IMEs) built with the meta tooling suite CINCO [31]. As we will see in the following, its graphical Domain Specific Language is a subset of the C/C++ language, and its code generator generates C/C++ source code from application models in this DSL. The specific hardware security we include in C-IME is based on SEcube security technology [5, 38], an open security platform that provides encryption running on a separate SoC hardware device.

Figure 1 shows the structure and functioning of C-IME: application models (top) are transformed to running C/C++ applications (bottom) by a model-to-code generator G (in the middle). Key to our approach is C-IME's dedicated graphical modelling language for model-to-code generators. This language is a Domain Specific Modelling Languages (DSMLs) [37], and it includes palettes of primitives that pertain to the application domain(s) its models cover. To address the specific holistic security language as an aspect, C-IME's own DSML collection of palettes is enhanced with a new palette of security primitives, populated with primitives implemented on top of the SEcube APIs (cf. Fig. 3). The model-to-code generator generates code from its argument application models fully automatically. Its structure is simple and consists of three parts, each designed in a dedicated model (cf. Sect. 5 and Fig. 1 (left)):

1. An **initialization model** generates setup code for the target platform. In our case study, the initialization concerns locating and setting up the SEcube device and performing the login procedure. In general, initialization can include any setup that is required to enable the inclusion of security features on a specific target platform.
2. The core of the **code generation** for the application model. The DSML for code generators defines the primitives needed to map the model primitives to their implementation level. In our case study, we use a DSML that corresponds to language constructs in C, and the model-to-code transformation generates C code from the models. In case we wish to target a different C platform or a different language, we need different implementations for the model's primitives, but we may start from the very same model. In our case study, this mapping is where we choose to target either a standard implementation for file operations, or a secure version that is based on SEcube based security technology.

¹We call IMEs the integrated development environments for (graphical) modelling languages, in analogy to the IDEs (integrated development environments) for classical code-level programming like Eclipse or IntelliJ.

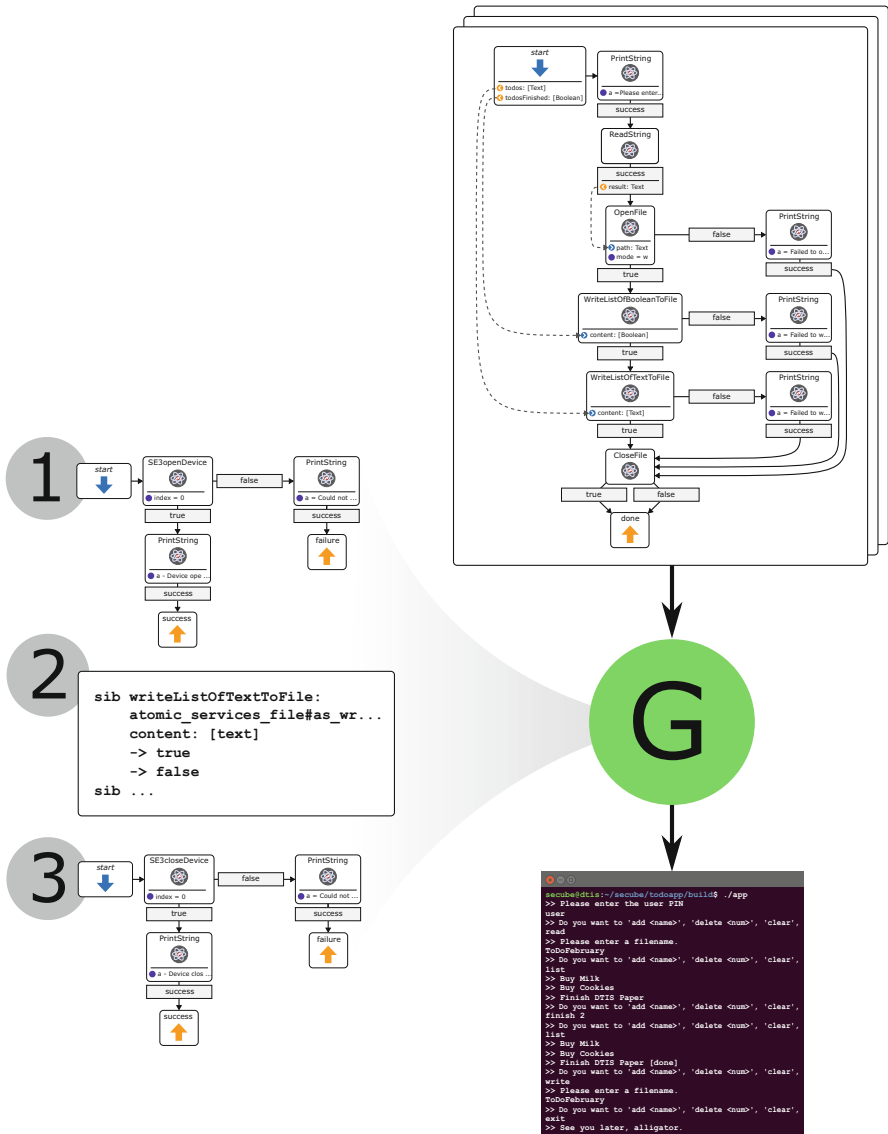


Fig. 1 The modelled code generator consists of three parts and generates applications fully automatically from process models

3. An **epilogue**, a counterpart model to the first phase. It generates code to clean up any setting or configuration due to security before the application terminates, for example logging out from the SEcube device. In general this epilogue includes any process needed to restore the original status of the environment and to enable the application to be executed again at a later point in time.

The case study illustrates how easy it is to model the code generator for a secure to-do list management application and how this generator differs from its insecure counterpart. The generated application comprises secure file operations to allow for the use in potentially insecure environments for data storage. This code generator can be used to secure the file handling of any application modelled in C-IME, not just the one in this case study. This specific model-driven approach is very flexible: the choice of a security implementation is configurable in the code generator. The SEcube security technology is our specific choice in this project, but any other implementation, be it in hardware, software or mixed, could be supported in an analogous fashion.

In Sects. 2 and 3 we introduce the reader to the C-IME modelling environment and the process of generating the code of applications from their models. Section 4 describes the SEcube hardware security technology and its API, and Sect. 5 shows how it is utilized in the case study. Our findings and an outlook on future work conclude in Sect. 6.

2 The C-IME Modelling Environment

C-IME is itself a product in a product-line of application modelling environments created with CINCO [31]. While C-IME targets headless C code applications running in a shell and is rather simple, other products in this IME-product line are more complex. For example, DIME (*Dynamic Web Application Integrated Modelling Environment*) is tailored to model web applications ready to be fully automatically code-generated and deployed into a Java EE web stack. Once the web application model is ready and the runtime environment is set up, the DIME generator produces and deploys a live online application literally with one click. For example, the website of the SEcube online community is completely modelled in DIME [4] and produced and maintained in this way. The existence of models allows verification of properties at the model level [11, 12], i.e. before code generation and independently of the runtime environment of choice.

This ease of handling and agility is possible because of the thoroughly model-driven approach to the design of IMEs embodied in CINCO. CINCO-products are based on meta models for graphs that are customizable to specific graph-like modelling languages by defining a set of modelling language specific *node* and *edge* types. The appearance of nodes and edges is associated with pictograms and properties of choice, to match style (representation look) and characteristics of the elements of a specific modelling language. These can be completely new modelling languages, but also well known ones: For example DFAs, Petri Nets, activity diagrams, BPMN were all easily modelled as CINCO model types [37]. CINCO further allows to define constraints on the graph structure that express further syntactic and behavioural properties of the models in the modelling language under definition. Examples are constraints on the nodes' connectivity (e.g., places can only be connected to transitions in Petri Nets, and viceversa), or on the number of

nodes of a certain kind in a model (e.g., there can only be a start SIB in Service Logic Graphs). In CINCO, the meta model of a specific graph language defines its syntax and properties, while its semantics are implemented in the associated code generator. A typical CINCO-product, therefore, is de facto a syntax aware editor for a target modelling language that comes with a code generator that takes models in that language and generates code for an execution platform of choice. In the case of C-IME, the target modelling language is Service Logic Graphs (SLGs, [27]), the domain of discourse is C applications, and the target code is a native (C) application that can run on a given target platform (a C runtime).

Practically, to create a domain specific modelling environment in CINCO an IME-designer chooses the modelling style of the CINCO-product under design – a sort of host model type – then associates with its graphical elements a set of primitives for a specific application domain, and then a model-to-code compiler that makes the generated code executable on a technology stack of choice. For example, in a different case study we chose again SLGs as host modelling language, but robotics as an application domain, and the ROS platform as target runtime platform [11, 12].

Here we use *Service Logic Graphs* (SLGs) as process models, and reuse the SLG definition of DIME, so both C-IME and DIME are close CINCO-products. Figure 2 shows a SLG belonging to the to-do list management application, the case study chosen to illustrate our security weaving concept. In SLGs, nodes on the graph canvas represent activities and different types of edges model control flow and data flow.

The activity nodes, e.g., `PrintString` in Fig. 2, are called *Service Independent Building Blocks* (SIBs). Analogously to a function and its parameters in general-purpose programming languages, a SIB represents a functionality and has a list of typed *input ports* (cf. `todos` and `todosFinished` nodes in the SIB labelled `start`) that provide the input data. Execution outcomes leading to distinct continuations are modelled using the concept of SIB *branches*, used to model the control flow of the system (cf. solid edges in Fig. 2). SIB branches are represented as branch nodes and are conceptually a constituent part of the SIB. In Fig. 2 we see `true` and `false` branches. Depending on the execution outcome of a SIB, the appropriate branch is followed, leading to the appropriate successor SIB, to be executed next. Outputs are provided by typed *output ports* that are located in the branches: the kind of output depends on the execution outcome, thus it makes sense to locate on each branch the output ports relevant to its continuation. E.g., there may be no computation result in error cases, thus error branches may not provide outputs, and different results can be associated with different output data.

The *start SIB* is the unique entry point of an SLG, represented by a dedicated node type labelled `start`. As the entry point is unique, a constraint expressing that there can only exist one start SIB in every model is enforced by the C-IME graphical editor. An SLG has one or more exit points, the *end SIBs* like the node labelled `done` in Fig. 2. Each end SIB defines a case of outcome of the entire process. Together, start and end SIBs with their branches and input/output types declare the interface of a process model. Consequently, a process model may be integrated into another SLG as a *process SIB* with input ports, branches and output ports (i.e., the actual

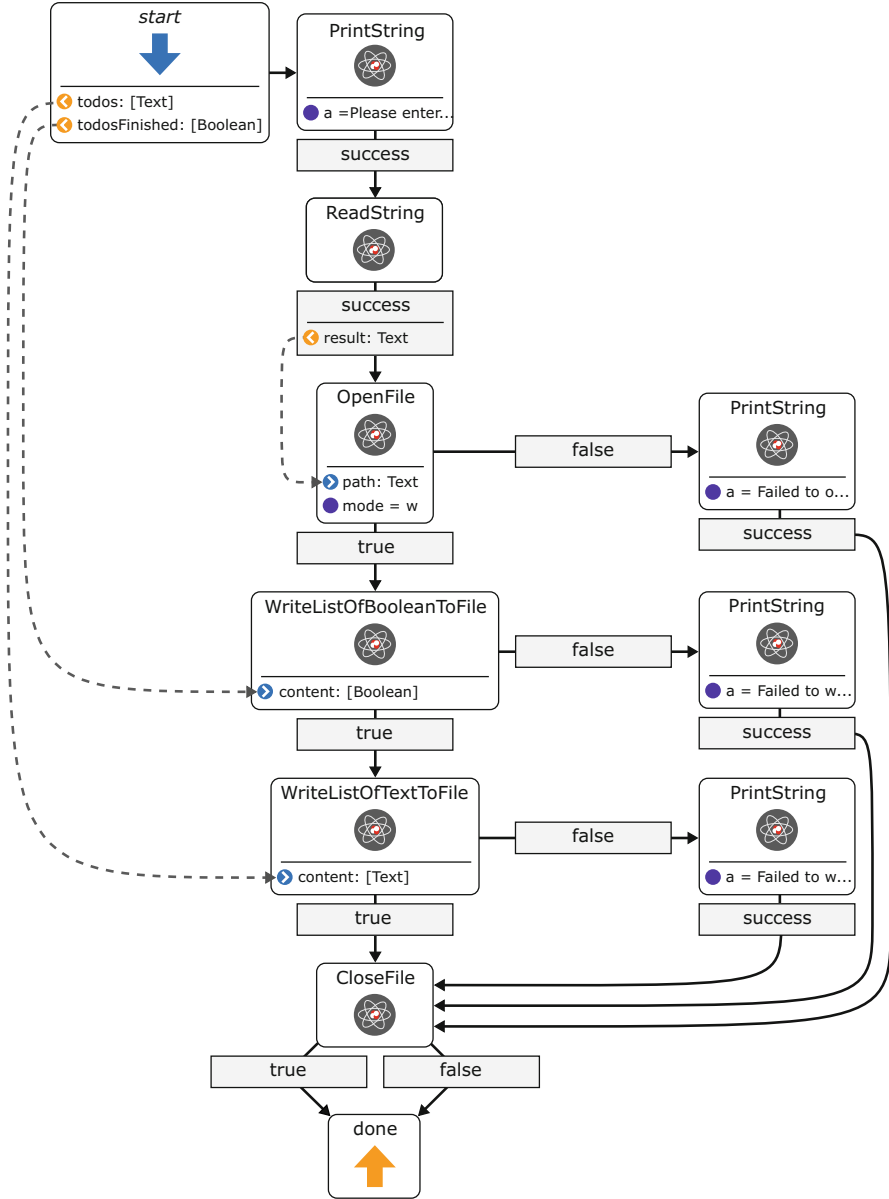


Fig. 2 C-IME model *WriteList* for writing the to-do list management application's data model to the file system

parameters) corresponding to the declaration in the sub process' start and end SIBs, thus introducing hierarchical, service-oriented modelling.

The primitive SIBs that define an application domain and all models of that domain ultimately base on are *native SIBs*. Here they represent calls to functions in C, which is the target programming language of C-IME's generator. In general *native SIBs* represent atomic entities that are implemented in some target programming language. The mapping between SIB interfaces and functions is done in a separate textual *domain-specific language*. For every SIB this language defines

- the SIB's name,
- its implementation as a reference to a file and function name,
- the input ports or function parameters
- and its branches with their respective output ports.

The following example is an excerpt from the SIB definition of the SIB `writeListOfTextToFile` used for the insecure version of the code generator:

```
sib writeListOfTextToFile:
  atomic_services_file#as_write_list_of_text_to_file
  content: [text]
  -> true
  -> false
```

The control flow is modelled via the solid arrows that connect a SIB node to its branches and these to the successor SIBs.

The data flow is modelled using dashed edges. Data can be supplied to a SIB by the outcome of a previously executed SIB within the same SLG or, for process SIBs, by their context through the initial output ports of their start SIB. Graphically, the data flow can be represented as

- *Direct data flow*: An output port may be directly connected to one or more input ports (cf. the dashed grey edge from output port `todos` to input port `content` in Fig. 2). Because the source of the data must be unique, every input port can get its direct data only from one output port, respectively one data source.
- *Variable in the data context*: if the data collection needed by a SIB comes from more than one SIB, the data source must be a variable in C-IME. Variables are represented as nodes, are used to store data and to provide data, and provide the means for an $n : m$ connection between output and input ports.

In both cases the types of the output ports and the connected input ports must match. The types are denoted after the name of a port (e.g., `todos: [Text]`). Square brackets around a type name denote a list type. A C-IME model validation component performs type checking and immediately indicates problems to the modeller [3]. C-IME supports the *primitive types* `Text`, `Boolean`, `Integer`, `Real`, `Timestamp` as well as their *list types*. For input ports with primitive types a modeller may supply a static value instead of using data flow edges (cf. the parameter `a` of the SIBs `PrintString` in Fig. 2).

3 Model-to-Code Generation in C-IME

A modeller in C-IME does not deal with details of the underlying implementation: the code generator takes care of transferring the semantics of the process models to a complete implementation in C code. The central artifact for this mapping is the library of implementations of the SIBs. The code generator walks along the structure of an application's model, and assembles its code using the applications's structure and the code implementing the SIBs. This library provides C counterparts for the types in C-IME as well as their list types. It also offers a simple memory management solution facilitating reference counting, shipped with C-IME by means of an easy to use project template. The code generation produces a fully functional C application that can be directly compiled and used.

The applications will be in many cases embedded systems, where target environments typically have constraint hardware, limited computing power and small available memory. Therefore, the code library implemented along with the generator is particularly lightweight.

Following the separate compilation principle [1], the C-IME code generator generates one *.c and one *.h file per SLG. Analogously, the library code and all the method calls for native SIBs base solely on the corresponding header files. This way, the implementation of all SIBs (including process SIBs) and library functions can be replaced transparently (cf. Sect. 5).

Each process model is generated separately, so it is also possible to use C-IME to model only parts of another application. The generated code is an implementation in C and can be seamlessly used in other C and C++ applications. This way the model-driven approach can integrate easily also into pre-existing non model-driven projects.

In contrast to Java – the generation target of DIME – C does not support high-level language features like object-orientation, subtype polymorphism, garbage collection or even namespaces. Hence, all this is emulated in the generated code, and it is dealt with by the code generator, hiding all this complexity from the modeller. For example, due to the global namespace in C the generator ensures uniqueness of all function names it generates by using the model name as a prefix, effectively emulating namespaces not supported by the target programming language.

The graph structure of a process model cannot be represented canonically in a block-oriented language like C [10]. This is solved by iterating through an adjacency structure in the generated code, an implementation that avoids deep call stacks. While the Java generator in DIME realizes SIB execution via calls to polymorphic `execute-methods`, this is emulated in C with function pointers to corresponding `execute-functions`. The generator renders dedicated implementations of these `execute-functions` for the various kinds of SIBs, which encapsulate the concrete execution as well as the data flow handling.

Every process is generated as a C implementation comprising

- a global function to initialize all the data structures needed for the process' execution,
- a global function implementing the control flow to execute the process and its SIBs, and
- a global function to explicitly free the process' data structures if the system runs short on memory. This feature aims mainly at embedded systems where memory resources may be limited.

The data flow is realized using a struct that is passed between SIB executions. Branches of an SLG are implemented as a struct with one member indicating the branch that was taken and one member for every branch holding an inner struct representing its output ports' values. All necessary calls to library methods for the creation of constants and lists, for memory allocations and deallocations as well as for reference counting are generated directly into the C code.

4 Hardware-Based Holistic Security

While software implementations of encryption are cost-effective and flexible, implementation in hardware is generally regarded as being more secure. It is less vulnerable to common attacks, of which a good summary can be found in [35], also discussing general advantages and disadvantages of software vs. hardware implementation of encryption (Fig. 3).

4.1 *SEcube Security Platform*

One example for a hardware security platform is the SEcube [38]. As shown in Fig. 4, the SEcube consists of three hardware components embedded in a single 9 mm × 9 mm BGA package:

- A powerful ARM Cortex M4, Floating Point, Low Power CPU,
- A flexible and fast FPGA for HW custom developments,
- An EAL5+ certified secure element.

Having these three elements together in a single SoC makes it possible to create a versatile and Common Criteria certified security environment where developers can rapidly implement very complex functions.

The platform implements a set of commonly used encryption algorithms that run on a device separate from the host machine. In this way the encryption mechanism is less vulnerable to the attacks listed in [35] than implementations in software. In particular, the following properties tackle common security risks:

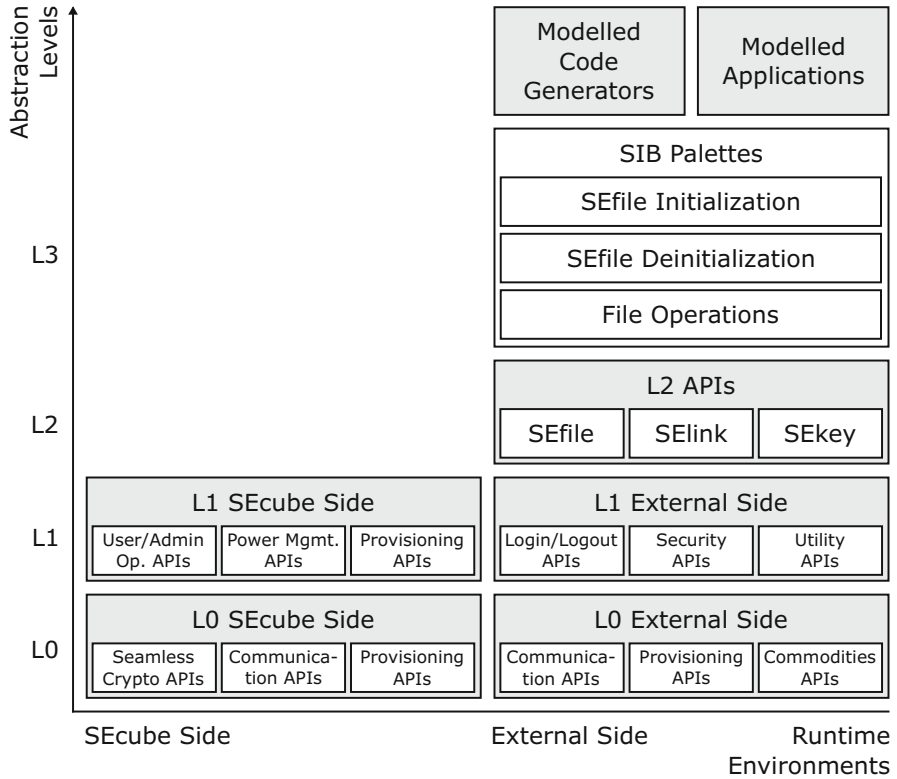


Fig. 3 Overview of the SEcube software collection [13]

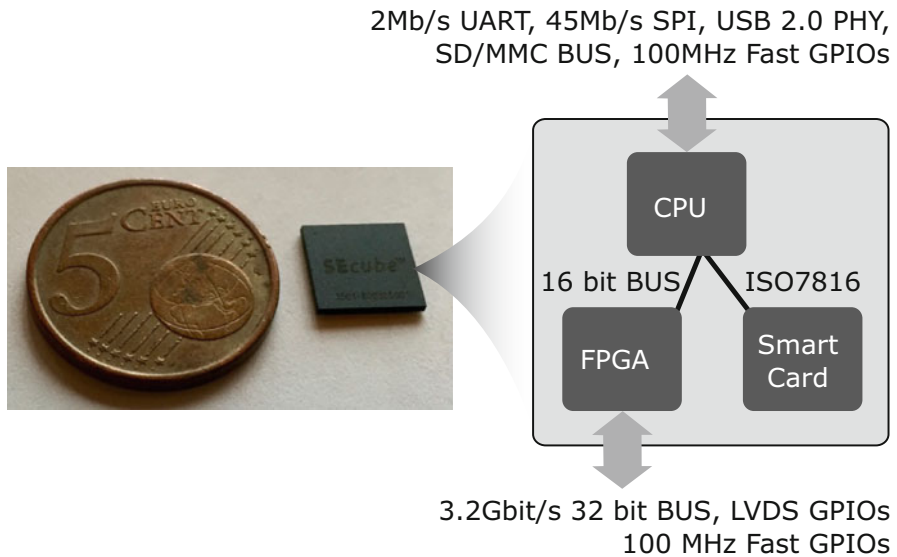


Fig. 4 SEcube hardware architecture [38]

- *No Security Bound Imposed by the OS*: All encryption algorithms run on the SEcube device, that is separate from the host machine. This way they rely in no way on host machine’s operating system and cannot be attacked through its potential security flaws.
- *No Arbitrary Memory Access*: The SEcube device has its own memory, that is physically separate from the host machine. It is therefore not accessible to other software running on the same machine.
- *Data Integrity Guarantee*: The firmware running on the SEcube device cannot be altered through its interfaces. While this makes the device more secure, it also makes the solution less flexible. Unlike software implementations of encryption, the SEcube firmware cannot be updated after the device was physically closed.
- *Secure Key Storage*: It is crucial to store keys used for encryption and decryption in a secure location. In case of the SEcube, keys are securely stored on the device which they will never leave. Keys are identified by their ID and can be used on the particular SEcube device only.

The SEcube platform is of particular interest to us because it is an entirely open platform. Most importantly, the software running on the device as well as the API to be included in applications on the host machine is open source. While the open source concept is common to software, it is not yet commonly applied to hardware especially in the security domain [38].

4.2 SEcube API

From the point of view of a software developer, the SEcube comes with a set of APIs for the programming language C. It is organized in three layers, allowing security experts to interact directly with the device drivers in a fine granular fashion, whereas non-experts, typically application developers who are security consumers, can use higher abstraction levels that shield most technical details. On the highest level, the API allows to interact with the file system and have all content implicitly encrypted. With this capability, user can take advantage of the technology without having to struggle with the details of encryption.

L0 API The lowest layer L0 of the API allows direct interaction with the SEcube device. It provides basic functionality to discover devices, to open and close them and also to communicate with them through one generic function. While some functions are to be used by experts only, device discovery and open/close operations are useful to every user. In our model driven approach, the API is servified, i.e. transformed (wrapped) into a collection of microservices that constitutes the L0 SIB palette. We provide as SIBs the most important functions of this L0 API layer, i.e., those needed to design the model-driven code generator that in our case study automatically adds security to the target application (cf. Sect. 5)

- `L0_discover_init` begins an iteration through all connected SEcube devices,

- `L0_discover_next` continues the iteration,
- `L0_open` opens a connected SEcube device, and
- `L0_close` closes an open SEcube device.

L1 API The second layer L1 of the API provides functionality to log into the device and to perform encryption and decryption of data on a byte level. This layer also allows setting up a connected SEcube device, meaning to add, delete and list keys, to set the user and the admin PIN and to list available algorithms. Functions of this layer that will occur in the case study are

- `L1_login` to authenticate as authorized user to an open SEcube device,
- `L1_logout` to terminate the session and to restore the device such that it accepts logins in the future, and
- `L1_crypto_set_time` to synchronize the time between the host machine and the SEcube device.

These functions are still close to the management of the security platform, not to operations on data. We find the corresponding SIBs in the models of the security-enhancing code generator.

L2 API The highest layer L2 of the API consists of the SEfile and SELink libraries, which are used to encrypt data at rest respectively data in motion. The provided functions implement common interfaces used to interact with the file system (SEfile) and to send data over networks (SElink). Encryption happens implicitly, so that the user does not have to juggle with technical details of the SEcube. In our case study, we utilize these libraries to automatically secure file operations of an application modelled in C-IME. In particular, we use the functions

- `secure_init` to initialize a SEfile session,
- `secure_open` to open a new SEfile,
- `secure_write` to encrypt and write a byte sequence to an open SEfile,
- `secure_read` to read and decrypt a byte sequence from an open SEfile,
- `secure_close` to close an open SEfile, and
- `secure_finit` to terminate a SEfile session.

5 Case Study: A To-Do List Management Application

To show the power and the flexibility of extreme model-driven design, we study the case of a small to-do list management application that allows users to keep track of activities to be done in the future. The application allows the user to add new items to a list and to keep track of their status, namely whether a task is finished or still pending. For convenience, we want to store the list in a cloud storage service, in order to access it from any location. While this application has a relatively small functionality, it has a high demand for security. Users want to securely save their to-do list in the potentially insecure cloud storage service. Nevertheless they wish

integrity: the list should not be accessible (readable or alterable) by unauthorized others including the cloud storage provider itself. More abstractly, this is at the same time a use case for many IoT and CPS applications, where a given data set should be prevented from unauthorized access (for tampering or reading) and securely transferred to a destination storage.

To securely save all data, we want to protect it by encryption before saving and uploading. Using C-IME, we can tackle the security cross-cutting concern in the code generator rather than in the application model. We thus model the application independently from our security requirement, so that the application model only concerns the pure functionality, remaining small and easy to comprehend and modify. All the file operations occurring in the model can be secured automatically by adding the security aspect per model-to-code transformation, during the ensuing code generation. In this way, we ensure that the security enhancement happens systematically, uniformly and coherently. The generated application will have all its file-related operations secured so that its files can be synchronized to cloud storage services at no risk.

The original to-do list management application was indeed already modelled in C-IME prior to the work on security. Like many other applications, it uses read and write file operations to persist its data model to the hard drive. These operations are typically not encrypted. In order to secure these operations we could have manually changed the model, adding the necessary operations for managing the SEcube hardware devices, the login process, the keys and encryptions/decryptions.

Much more desirable would be to use a technique that does not require any change of the application's models, and sufficiently generic for the models and applications that the very same technique can be applied systematically to any other application modelled in C-IME. To this aim, we propose a method that instead models the C-IME code generator using yet other domain-specific languages in such a way that critical parts of any C-IME-modelled application can be automatically secured during its code generation as part of a security-enhanced code generation process.

The resulting to-do list management application is a near-hardware application in C/C++ that interacts with the SEcube through its C API. As such it is ideal to study the appropriateness of the extreme model-driven approach implemented in C-IME when applied to a low-level target programming language and when interacting with hardware.

5.1 The To-Do List Management Application

The basic functionality of a to-do list management application is generally known, but we will describe the features included in this case study in detail in order to then discuss the security critical parts of the actual C-IME models. The fully generated to-do list management application is a command line tool that allows a user to manage all the items on a to-do list by means of simple commands. The application further allows the user to write the list to a file and read it back. This can be done

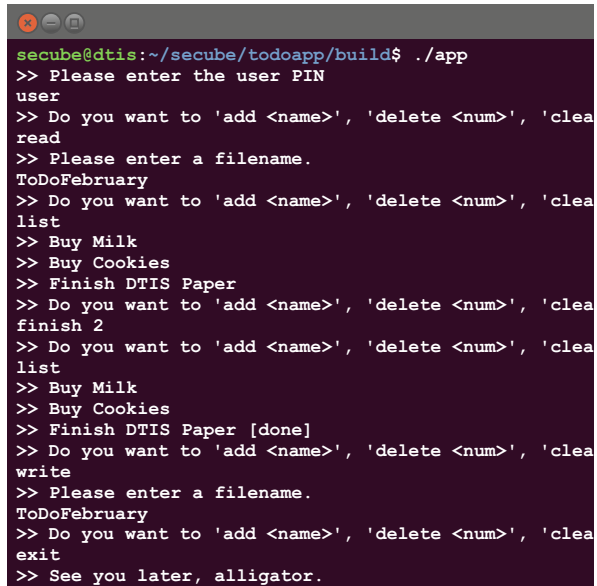
in a secure manner with encryption using the SEcube technology or in an insecure manner using standard C functionality. Both alternatives will be generated from the very same model, allowing the user to choose at generation time the preferred one.

The to-do list management application takes input commands and processes them in a loop until the user terminates the application. The application offers the following user commands:

- `add title` adds a new element with a given title to the to-do list. The element is initially added as unfinished.
- `finish i` sets the status of the i th element to finished.
- `delete i` deletes the i th element from the to-do list counting from zero.
- `clear` deletes all items from the to-do list.
- `list` prints all items and their status to the console.
- `write` allows the user to write the to-do list to a file. The user is prompted for a filename. In the secure version of the to-do list management application this operation will encrypt all data so that all files can be stored in cloud storage where they are unreadable even to other parties that have access to the storage.
- `read` allows to read a to-do list from a file. The user is prompted for a filename and all the items are appended to the existing to-do list. For the secure version of the to-do list management application this operation also decrypts the data to make it readable to the user.
- `exit` terminates the application.

Figure 5 shows a screenshot of the to-do list management application during execution. In this secure version, the user is prompted for the PIN to log into the

Fig. 5 Secure to-do list management application during execution



```

secube@dtis:~/secube/todoapp/build$ ./app
>> Please enter the user PIN
user
>> Do you want to 'add <name>', 'delete <num>', 'clear', 'read'
read
>> Please enter a filename.
ToDoFebruary
>> Do you want to 'add <name>', 'delete <num>', 'clear', 'list'
list
>> Buy Milk
>> Buy Cookies
>> Finish DTIS Paper
>> Do you want to 'add <name>', 'delete <num>', 'clear', 'finish 2'
finish 2
>> Do you want to 'add <name>', 'delete <num>', 'clear', 'list'
list
>> Buy Milk
>> Buy Cookies
>> Finish DTIS Paper [done]
>> Do you want to 'add <name>', 'delete <num>', 'clear', 'write'
write
>> Please enter a filename.
ToDoFebruary
>> Do you want to 'add <name>', 'delete <num>', 'clear', 'exit'
exit
>> See you later, alligator.

```

SEcube immediately after the application is started. The user reads a to-do list from a file, marks one item as finished and writes the list back to the file.

5.2 *To-Do List Management Application Model*

The to-do list management application already modelled in C-IME prior to this work was security-agnostic and it did not use encryption. The desire to retrofit security on the app arose with this work, and the security requirement to encrypt all data when stored on the hard drive (but it could also happen in the cloud) was stated only after the original application was deployed and running.

Identifying the security critical operations with regard to this security requirement is easy as all C-IME applications share the same dedicated SIB palette. For the to-do list management application, the critical operations are `OpenFile`, `Write...ToFile`, `Read...FromFile` and `CloseFile`. As the C-IME models for reading and writing the to-do list are similar, in the following we will describe the write process.

Figure 2 shows the model *WriteList* within the to-do list management application that writes its simple data model to the hard drive. The only data stored in memory at runtime are the to-do list's items. They consist of a title and a Boolean state indicating whether or not the item was already marked as finished. In the application's data model this information is represented by a list of texts and a list of Boolean values. Only these two lists are passed to the write process as input parameters `todos` and `todosFinished`.

Upon execution start of this model, the first SIB executed, `PrintString`, prompts the user to enter a filename that identifies which file to write to. The SIB receives a static text as an input and prints it to the console telling the user what to do. This SIB has only the default `success` branch and no data output. The SIB `ReadString` reads the user's choice from the command line. This SIB too has only the default `success` branch, but it provides the desired filename as a result through an output port. With this information, the successive SIB `OpenFile` can open the file specified by the filename input directly connected to the output port of its predecessor. File operations can fail for many reasons, so a Boolean return value indicates whether or not the operation was successful. Accordingly, this C-IME SIB has the two branches `true` and `false`, and the control flow is split in two alternative continuations. In case of success, the model proceeds to write the data model to the open file, first the list of booleans, then the list of data. In case of failure it prints an error message and aborts the process. If all file operations succeed, the execution traces vertically from the top down, opening the file, writing both lists to it and finally closing it. The lists are passed from the sub models input parameters to the SIB's input ports through direct data flow edges. If any file operations fails, the model reaches the corresponding `PrintString` SIB, which prints a meaningful static error message before the sub model terminates.

Both read and write file operations are available for all the primitive types and lists supported by C-IME, allowing the modeller to seamlessly persist the to-do list management application's data model to the file system.

5.3 *Modelling a Secure Code Generator*

The SIBs used in the to-do list management application can be linked to one or more implementations conform to their respective interfaces (cf. Sect. 3). By changing the implementation to which a SIB is associated, it is possible to generate different versions of an application from the exact same C-IME model. In particular, using customized code generators this allows us to generate both a secure and an insecure version of the to-do list management application from its model. They share most of the SIB definitions, only the file-related operations `OpenFile`, `Read...FromFile`, `Write...ToFile` and `CloseFile` have to be generated in the two versions with different implementations.

As described in Sect. 3, in our service-oriented approach implementations of native SIBs are invoked from within the generated code. The code of the expected implementations must therefore be available when compiling the generated code. It is at this point that a customized code generator may choose one of multiple implementations to automatically secure file operations. Such a customized code generator is independent of the specific application model and can thus be used to secure not only the to-do list management application but any application modelled in C-IME.

To use the hardware encryption provided by the SEcube we also need to initialize the SEcube API and to close it. These steps can be routinely performed at the very beginning respectively at the very end of any application's execution, so that we decided to make them part of the enhanced code generator itself. In this way the actual application's models remain untouched and security requirements are automatically implemented by the code generation process.

Accordingly, to model a code generator for C-IME we need additional domain-specific languages and models to define the following three parts:

- An initialization process to set up the environment for the SIB implementations that are included in the generated application (cf. Part 1 in Fig. 1). For the SEcube this is the login routine, where the device is opened and the user authenticated. The initialization process is modelled using SLGs similarly to the main application, but with a different SIB palette dedicated to initialize the SEcube. The modelled process will be automatically included in any generated application by this particular code generator.
- A mapping from SIBs in the model to concrete implementations written in C (cf. Part 2 in Fig. 1). Depending on this mapping, the code generator will choose the secure or the insecure version of file operations.

- A finalization process, as the counterpart to the initialization to clean up the environment (cf. Part 3 in Fig. 1). The logout procedure from the SEcube device happens here.

These three parts together model the code generator that can then be used to generate the to-do list management application fully automatically from its models. Figure 1 visualizes both, the code generator models and the application models. The three parts on the left side, initialization, SIB mapping and finalization define the code generator G. The generator G is then used to generate the concrete application in the desired version with its desired properties fully automatically from the application models. The result is a C implementation of to-do list management application that can be compiled and used from the command line.

For the to-do list management application, the standard and secure versions of the application's code are generated by two different code generators, both modelled in our domain-specific languages. They share the same mapping from SIBs to implementations, except for the file operations: the standard code generator includes an implementation based on the standard C file operations, while the code generator for secure applications includes an implementation based on the SEfile API. As a policy, only the qualifier of the function name would differ in the code generators, while the SIBs signature is the same. In the code generator of secure version of the apps, for the `writeListOfTextToFile`-SIB this mapping is defined as follows:

```
sib writeListOfTextToFile:
  atomic_services_se3#as_se3_write_list_of_text_to_sefile
  content: [text]
  -> true
  -> false
```

For the insecure version if the code generator only the function name differs while the remainder of the SIB definition equals.

Initialization and finalization are only needed to set up the SEcube environment. The code generator for secure apps thus includes the modelled login and logout processes, while for the standard code generator these two processes are empty.

Figure 6 shows the model for the secure code generator's initialization. The model invokes a sequence of native SIBs each of which can succeed or fail. In case of failure an error message is printed to inform the user and the model immediately terminates. To enable the SEfile API used by the SIB's secure implementations, the SEcube is opened, the user is prompted for the PIN and the login is finally performed. To use encryption it is further necessary to set the time and encryption parameters. The model for finalization is even simpler, keeping all models for the secure code generator small and simple.

The initialization and finalization models in C-IME use the SIB palette of SEcube technology security primitives. This palette comprises all the needed functionalities of the various SEcube APIs that are needed to allow for secure file operations, and lifts them to the C-IME modelling level. For our case study, we used the following security primitive SIBs:

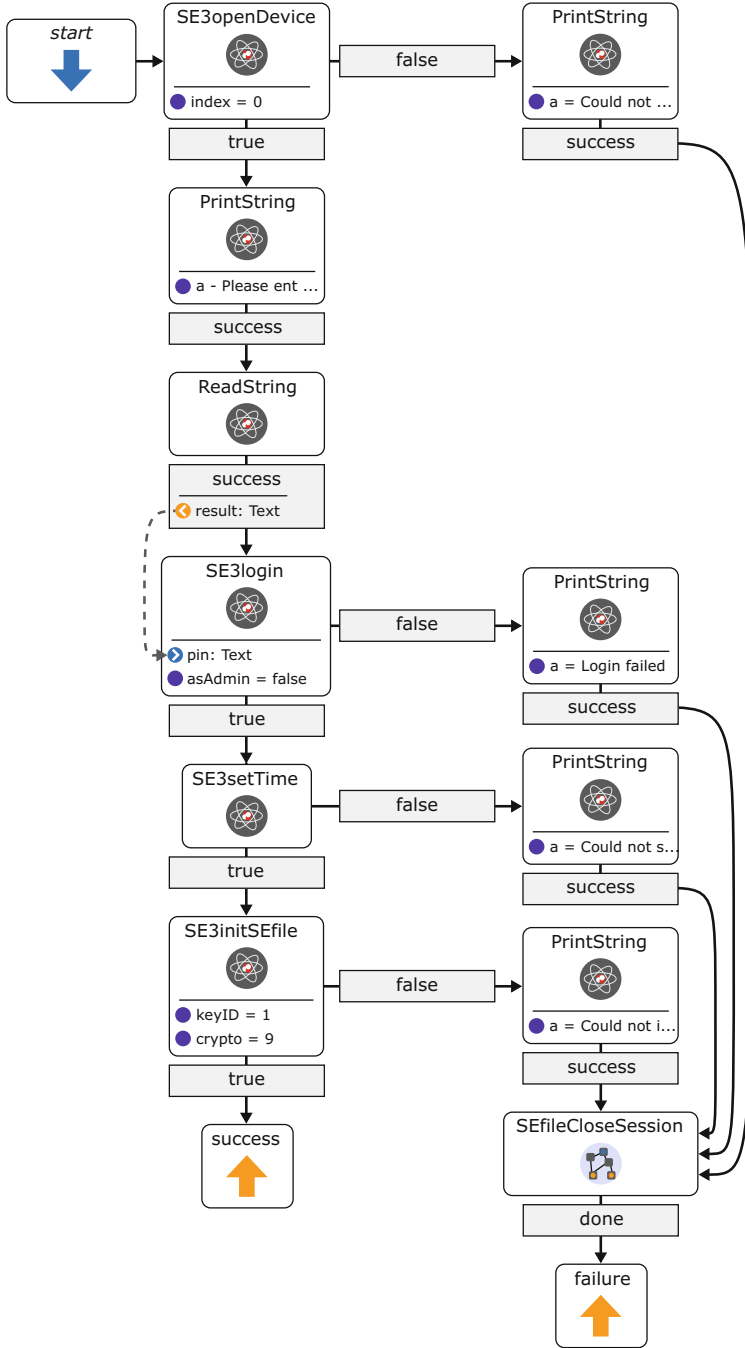


Fig. 6 C-IME model for setting up the SEcube environment to allow for secure file operations

- `SE3openDevice` allows to find and to open the i th connected SEcube device. This SIB uses internally the SEcube API functions `L0_discover_init` and `L0_discover_next` to iterate through connected devices and to open the chosen one. Only the integer i is expected as an input.
- `SE3login` is used to authenticate as authorized user to an open SEcube device. The SIB is based on the API function `L1_login`. It expects as an input a textual user PIN and the role of the user, namely whether or not admin privileges are sought.
- `SE3setTime` synchronizes the time between the host machine and the SEcube device. This synchronization is required before invoking any encryption algorithm. The implementation wraps the API function `L1_crypto_set_time`.
- `SE3initSEfile` initializes the SEfile session that is needed to enable successive secure file operations. The SIB is implemented based on the SEcube function `secure_init` and it uses default parameters for simplicity.
- `SE3finitSEfile` wraps the function `secure_finit` and closes the SEfile session.
- `SE3logout` is based on `L1_logout` and terminates the login session to restore the device status such that it can again accept logins.
- `SE3closeDevice` is based on `L0_close` and closes the device to allow for it to be reopened and used in the future.

All these SIBs have `true` and `false` branches, indicating whether or not the respective operation was successful. Figure 7 shows a screenshot of the secure code generator’s initialization process in C-IME. On the right side the palette of security primitives is highlighted in blue.

This way, the code generator is itself very flexible, and generates code for any application modelled in C-IME that has implicitly secured file operations. Note that the code generator is in no way specific to the to-do list management application but it can easily be applied to any other C-IME application as well. Because the code generator is itself modelled, it is also very flexible and extensible: we are not limited to using only SEcube security technology. In fact, we could wrap any other implementation for secured operations in an analogous fashion, extending the palette of microservices or creating a new palette, this way making other security mechanisms and technologies available to the application designers.

5.4 Ready for Cloud Storage

In many projects, data must typically be accessible not only to one person but to many members of a team, often in file form. Convenient and popular file sharing technologies are cloud storage services such as Dropbox [7], Google Drive [14] or OneDrive [34]. These services synchronize the local hard drive with the cloud service and make files and folders available on multiple machines to many users and different locations. In the context of smart spaces and Industry 4.0, this way of

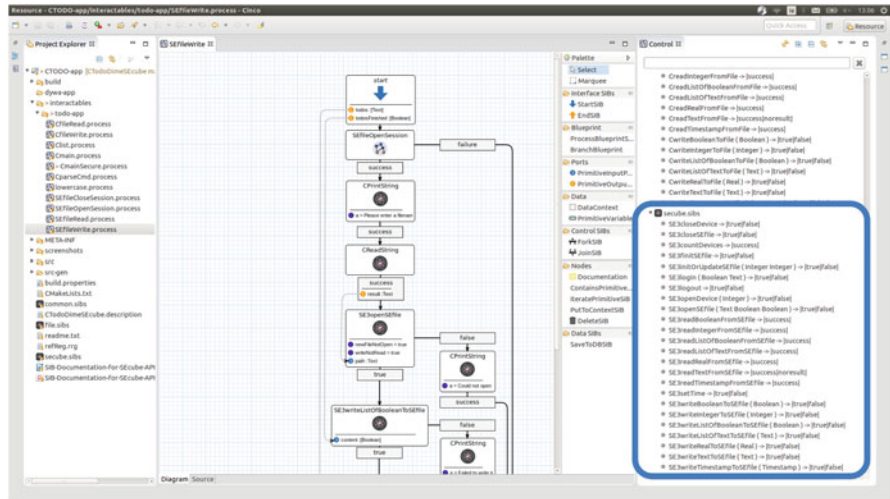


Fig. 7 Screenshot of the code generator’s initialization process in C-IME. The SIB palette is highlighted in blue

storing and potentially sharing data is seen as attractive, but with caveats: While this is a convenient solution that integrates well into the operating system’s file system, it presents data security and compliance problems. In numerous cases unauthorized parties gained access to data stored in the cloud [8]. Because the data is stored externally on servers under the control of third parties, there is no way to configure these services with regard to own custom security requirements. Rather than relying on the security of these cloud storage services, one can make the data that is being uploaded unusable to others. This can be achieved through encryption – ideally hardware encryption – to grant access only to users in possession of the physical encryption device that holds the correct key. In this way a high standard of security can be achieved with no assumptions about the underlying cloud storage technology. Even if the data was stored in a completely public manner, meaning that the general public was allowed to download the files, this approach would still guarantee high security.

Our case study showed how hardware security features based on SEcube technology can be seamlessly integrated into C-IME models and in the code of the applications. Applications secured in this way allow for the secure use of cloud storage services, in the sense that they rely only on the trust in the underlying encryption technology but do not rely on the environment. This way, we can free users from the need to trust the security of the cloud storage technology, so that they can create applications that are secure by construction in a fashion controlled by the application designer. This capability allows taking advantage of services that are either not secure, or whose security mechanisms are not configurable or controllable as a user, including existing potentially insecure cloud storage

technology. Following our model-driven approach to holistic security there is no need to trust in the security mechanisms of the underlying environment.

Our approach is flexible and can be used to security-enhance not only file operations but any operation that is modelled in C-IME. In this way our approach is not limited to deal with insecure file storage environments but it can also be used to base on insecure channels of communication. In the very same way we secured file operations during code generation, we can secure communication operations as well. All complexity of this is hidden from the modeller. He or she will work with “normal” communication primitives while the code generator takes care of the security fully automatically.

6 Conclusion

In this chapter we have shown a model-driven and generative approach to holistic security. We have shown how running C/C++ applications that were developed in a model-driven fashion can be easily enhanced with hardware secured file operations simply by changing the according code generator to make them ready for insecure environments. In this way we create applications that can store their data in potentially insecure cloud storage services. Key to our approach is the full code generation philosophy of XMDD, which guarantees that each running application has a valid model from which it was generated. This approach allows us to realize certain cross-cutting concerns like security via a corresponding code generator in a fashion reminiscent of aspect oriented programming [21] and without touching the application models. The code generator itself is modelled using a dedicated graphical modelling language for code generators that embodies a palette of security primitives based on any underlying hardware or software security technology. We have illustrated how easily such code generators can be modelled using a to-do list management application as a case study. It should be noted that handling security via the code generator this way is not application specific but allows us to automatically secure the file handling of any application that was modelled this way.

Currently, we only secure applications concerning data at rest. The philosophy underlying the XMDD paradigm [29], where complex applications are modelled via a number of strongly linked individual models is however much more general, as witnessed, e.g., by DIME [3, 5, 16] our most elaborate CINCO-product variant. We are currently investigating how to best approach also aspects like data in motion in this context. Additionally, up to now we introduced solely compile-time variability for C-IME, although we have shown the impact of runtime variability for a wide range of applications [32, 33] for high-level programming language targets. We are planning to apply this approach also to C-IME. The to-do list management application, e.g., would significantly benefit if the application could dynamically decide at runtime to use hardware security in case the SEcube device is available, and degrade to a software security variant as long as a cryptographic key is known, or else simply use an insecure variant. Important for us is, however, the simplicity

principle [25, 26], which clearly favours reduced tailored solutions to generic approaches. In this case this means for a modeller to be able to model any C-IME application without worrying about security concerns, which adds an incremental requirement specification flavour to the entire design approach [17]. This feature is of particular importance in the context of smart advanced manufacturing, as investigated by the new Irish Science and Research Centre Confirm. In that context, we plan to extend the work on aspect-enhancing code generation for a variety of platforms building on this specific experience and on the legacy of Genesys [19, 20], as well as using synthesis [23, 36] based on loose programming [22, 27] to provide correct-by-construction techniques to deal with security enhancement. The goal is to apply them seamlessly to robotics-like applications, as in [18, 24], and more recently in [11, 12].

Acknowledgements This work was supported, in part, by Science Foundation Ireland grant 13/RC/2094 and co-funded under the European Regional Development Fund through the Southern & Eastern Regional Operational Programme to Lero - the Irish Software Research Centre (www.lero.ie).

References

1. Ancona D, Lagorio G, Zucca E (2002) A formal framework for java separate compilation. In: ECOOP Proceedings, ECOOP'02. Springer, London, pp 609–636
2. Björck F, Henkel M, Stirna J, Zdravkovic J (2015) Cyber resilience – Fundamentals for a definition. Springer, Cham, pp 311–316
3. Boßelmann S, Frohme M, Kopetzki D, Lybecait M, Naujokat S, Neubauer J, Wirkner D, Zweihoff P, Steffen B (2016) Dime: a programming-less modeling environment for web applications. In: Proceedings of the ISO/ISA 16 Proceedings. LNCS, vol 9953. Springer, Cham, pp 809–832
4. Boßelmann S, Kühn D, Margaria T (2017) A fully model-based approach to the design of the secube™ community web app. In: Proceedings of the 2017 12th International Conference on Design Technology of Integrated Systems In Nanoscale Era (DTIS). IEEE, Piscataway, pp 1–7
5. Boßelmann S, Neubauer J, Naujokat S, Steffen B (2016) Model-driven design of secure high assurance systems: an introduction to the open platform from the user perspective. In: Margaria T, Solo MGA (eds) SAM'16. Special track “End-to-end Security and Cybersecurity: from the Hardware to Application”. CSREA Press, USA, pp 145–151
6. Devanbu PT, Stubblebine S (2000) Software engineering for security: a roadmap. In: FOSE Proceedings, ICSE'00. ACM, New York, pp 227–239
7. Dropbox. <https://www.dropbox.com>. Accessed 18 Nov 2017
8. Dropbox hack leads to leaking of 68m user passwords on the internet. <https://www.theguardian.com/technology/2016/aug/31/dropbox-hack-passwords-68m-data-breach>. Accessed 18 Nov 2017
9. Elahi G, Yu E, Li T, Liu L (2011) Security requirements engineering in the wild: a survey of common practices. In: Proceedings of the 2011 IEEE 35th COMPSAC. IEEE, Piscataway, pp 314–319
10. Engeler E (1971) Structure and meaning of elementary programs. In: Proceedings of the Symposium on Semantics of Algorithmic Languages. Springer, Berlin, pp 89–101
11. Farulla GA, Indaco M, Legay A, Margaria T (2016) Model driven design of secure properties for vision-based applications: a case study. In: Proceedings of the International Conference

- on Security and Management (SAM). World Congress in Computer Science Computer Engineering and Applied Computing (WorldComp). CSREA Press, USA, pp 1–6
12. Farulla GA, Lamprecht AL (2017) Model checking of security properties: a case study on human-robot interaction processes. In: Proceedings of the 2017 12th International Conference on Design Technology of Integrated Systems in Nanoscale Era (DTIS). IEEE, Piscataway, pp 1–6
 13. Farulla GA, Prinetto P, Varriale A (2017) Holistic security via complex HW/SW platforms. In: Proceedings of the 2017 12th International Conference on Design Technology of Integrated Systems in Nanoscale Era (DTIS). IEEE, Piscataway, pp 1–6
 14. Google Drive. <https://www.google.com/drive>. Accessed 18 Nov 2017
 15. Gossen F, Neubauer J, Steffen B (2017) Securing C/C++ applications with a secube™-based model-driven approach. In: Proceedings of the 2017 12th International Conference on Design Technology of Integrated Systems in Nanoscale Era (DTIS). IEEE, Piscataway, pp 1–7
 16. Gossen F, Tiziana M, Göke T (2016) Modelling the people recognition pipeline in access control systems. Proc Inst Syst Program RAS 28:205–220
 17. Jonsson B, Margaria T, Naeser G, Nyström J, Steffen B (2001) Incremental requirement specification for evolving systems. Nordic J Comput, 8(1):65–87
 18. Jorges S, Kubczak C, Pageau F, Margaria T (2007) Model driven design of reliable robot control programs using the jABC. In: Fourth IEEE International Workshop on Engineering of Autonomic and Autonomous Systems, EASe'07. IEEE, Piscataway, pp 137–148
 19. Jörges S, Lamprecht AL, Margaria T, Schaefer I, Steffen B (2012) A constraint-based variability modeling framework. Int J Softw Tools Technol Transfer, Springer, Berlin, Heidelberg, 14(5):511–530
 20. Jörges S, Margaria T, Steffen B (2008) Genesys: service-oriented construction of property conform code generators. Innov Syst Softw Eng 4(4):361–384
 21. Kiczales G, Lamping J, Mendhekar A, Maeda C, Lopes C, Loingtier JM, Irwin J (1997) Aspect-oriented programming. In: Akşit M, Matsuoka S (eds) ECOOP'97. LNCS, vol 1241. Springer, Berlin, pp 220–242
 22. Lamprecht AL, Naujokat S, Margaria T, Steffen B (2010) Synthesis-based loose programming. In: Proceedings of the 2010 Seventh International Conference on the Quality of Information and Communications Technology. IEEE, Piscataway, pp 262–267
 23. Lamprecht A, Steffen B, Margaria T (2016) Scientific workflows with the jABC framework – a review after a decade in the field. STTT 18(6):629–651
 24. Margaria T, Steffen B (2007) LTL guided planning: revisiting automatic tool composition in ETI. In: Proceedings of the 31st IEEE Software Engineering Workshop (SEW 2007). IEEE, Piscataway, pp 214–226
 25. Margaria T, Steffen B (2010) Simplicity as a driver for agile innovation. Computer 43(6):90–92
 26. Margaria T, Floyd BD, Steffen B (2011) IT simply works: simplicity and embedded systems design. In: Proceedings of the IEEE 35th COMPSACW. IEEE, Piscataway, pp 194–199
 27. Margaria T, Steffen B (2004) Lightweight coarse-grained coordination: a scalable system-level approach. STTT 5(2–3):107–123
 28. Margaria T, Steffen B (2008) Agile IT: thinking in user-centric models. In: Margaria T, Steffen B (eds) ISoLA'08 Proceedings. Springer, Berlin/Heidelberg, pp 490–502
 29. Margaria T, Steffen B (2009) Business process modelling in the jABC: the one-thing-approach. In: Cardoso J, van der Aalst W (eds) Handbook of research on business process modeling. IGI Global, Hershey
 30. Margaria T, Steffen B (2012) Service-orientation: conquering complexity with XMDD. In: Hinchey M, Coyle L (eds) Conquering complexity. Springer, London, pp 217–236
 31. Naujokat S, Lybecait M, Kopetzki D, Steffen B (2018) Cinco: a simplicity-driven approach to full generation of domain-specific graphical modeling tools. Int J Softw Tools Technol Transfer 20:327. <https://doi.org/10.1007/s10009-017-0453-6>
 32. Neubauer J, Steffen B (2013) Plug-and-play higher-order process integration. Computer 46(11):56–62

33. Neubauer J, Steffen B (2013) Second-order servification. In: Herzwurm G, Margaria T (eds) Software business. From physical products to software services and solutions. LNBIP, vol 150. Springer, Heidelberg, pp 13–25
34. Onedrive. <https://onedrive.live.com>. Accessed 18 Nov 2017
35. Sklavos N, Toulou K, Efstathiou C (2006) Exploiting cryptographic architectures over hardware vs. software implementations: advantages and trade-offs. In: Biolek D (ed) AEE'06 Proceedings, WSEAS. Stevens Point, Wisconsin, pp 147–151
36. Steffen B, Margaria T, Freitag B (1993) Module configuration by minimal model construction. Technical report, Technical Report MIP Technical Report MIP 9313, Fakultät für Mathematik und Informatik, Universität Passau
37. Steffen B, Naujokat S (2016) Archimedean points: the essence for mastering change. *Trans Found Mastering Change* 1:22–46
38. Varriale A, Vatajelu EI, Natale GD, Prinetto P, Trotta P, Margaria T (2016) Secube™: an open-source security platform in a single SOC. In: DTIS Proceedings. IEEE, Piscataway, pp 1–6

Part III
Industrial Applications

Multi-range Decoy I/O Defense of Electrical Substations Against Industrial Control System Malware



Julian L. Rrushi

Abstract Industrial control system malware campaigns, such as BlackEnergy and Dragonfly, targeted electrical substations at various ranges relative to the computers that pushed the attacks into substation relays after being infected. Worm-like propagation of industrial control system malware in the Internet traverses paths along computers that may be far from their target, and that often are completely unrelated to power grid functions. Industrial control system malware hop from computer to computer until landing on one that has access to a target industrial environment. Industrial control system malware enabled by spear-phishing or website redirection attacks exploit web browser vulnerabilities coupled with human factors of energy company personnel. Watering hole attacks cause the installation of industrial control system malware on the computers of power grid operators, and sometimes even on the protective relays of an electrical substation. In this chapter we present a line of work that creates and operates industrial mirages, i.e., phantom substation targets for industrial control system malware to pursue, to intercept such malware bound for the power grid. The discussion focuses on decoy I/O. We also generally describe other key elements of industrial mirage at large, and explain how decoy I/O and those elements work together as integral components of the industrial mirage capability. Industrial mirage is able to actively redirect industrial control system malware to decoys, and can sustain prolonged interaction with such malware. We validated this line of work against numerous malware samples involved in recent industrial control system malware campaigns.

Keywords Industrial control systems · Malware · Defensive deception.

J. L. Rrushi (✉)

Department of Computer Science, Western Washington University, Bellingham, WA, USA
e-mail: julian.rrushi@wwu.edu

1 Introduction

The electrical power grid was recently targeted by industrial control system (ICS) malware during the BlackEnergy [1, 2] and Dragonfly [3] campaigns. With ICS malware we mean malware that contains code that is specific to industrial control systems. The BlackEnergy and Dragonfly ICS malware exploited vulnerabilities in computers and human factors, and were able to self-propagate. It has been known to the cyber security community for quite some time that an attacker can program malware to cause physical destruction of power grid equipment, or silently conduct cyber espionage to prepare for one. Those are exactly the operations that were performed by BlackEnergy and Dragonfly, respectively. BlackEnergy enabled attackers to access the protective relays of electrical substations over the network. Those accesses allowed for operating circuit breakers to cause power outages to several large regions. Dragonfly did not attempt physical alterations of the electrical power grid, however it conducted extensive cyber espionage on every industrial computer it could compromise.

Contribution We discuss our work on stealth defensive deception to protect the electrical power grid from malware attacks. The cyber-physical system of reference is an electrical substation. Deception is commonly perceived as a characteristic of a dishonest person, but in reality that is not always the case. In times of conflict, deception can be a formidable defense tool. The use of deception for legitimate defense predates modern time. For example, a fifteenth century historical figure known as the hero of Europe [4], namely George Kastrioti, war name Skanderbeg, was able to defend his country from Ottoman conquest for 25 years. The Ottoman empire at that time was one of the most powerful military powers on earth, whereas Skanderbeg's army was small in size and drew its members mostly from the general population. Skanderbeg's use of smart deception techniques provided various advantages on the battlefield and in the background.

The contribution of this chapter is twofold. Firstly, we discuss developments of industrial mirage as a deception capability for defending the electrical power grid from ICS malware. Industrial mirage is not a standard term in industrial process control, but rather a term that we chose to dub the defensive deception work that we discuss in this chapter. This will be a general discussion given with an eye towards the future, and centered on areas that have shown potential during head-to-head runs against live ICS malware. Secondly, we give a detailed discussion of decoy I/O, which may be considered the foundation of industrial mirage. We also explain how decoy I/O supports and cooperates with the other elements of industrial mirage. Decoy I/O projects an industrial mirage at various ranges, enabling the defender to intercept ICS malware at multiple locations while en route from a compromised computer to a target electrical substation. In addition to intercepting ICS malware after the fact, decoy I/O and the other elements of industrial mirage work together to proactively interfere in the ICS malware's target selection, redirecting ICS malware to decoys.

Chapter organization The remaining of this chapter is organized as follows. Section 2 provides background on ICS malware campaigns and attack code. Section 2 discusses also the effectiveness of industrial honeypots versus those ICS malware. In Sect. 3 we discuss the prospect of industrial mirage. In Sect. 4 we describe the main ideas behind the decoy I/O concepts, as well as their core architectures and principal techniques. Section 5 provides a broad description of the other elements of industrial mirage, and explains their cooperation with decoy I/O. In Sect. 6 we discuss an empirical evaluation of the performance of industrial mirage in a testbed that resembles an electrical substation. In Sect. 7 we summarize our contribution, and conclude the chapter.

2 ICS Malware vs. Current Defensive Deception

ICS malware and attack code The ICS malware used in BlackEnergy and Dragonfly targeted electrical substations at various ranges. They penetrated the networks of energy companies through e-mail spear phishing attacks supported by spam campaigns. Phishing e-mails contained malicious attachments, such Portable Document Format (PDF) containing embedded JavaScript code, or Microsoft Office file attachments that leveraged the macro functionality. Both resulted in code execution on the target computer. The attackers ran watering hole attacks by injecting an HTML iframe into several websites related to energy. The HTML iframe redirected visitors to another website, which the attackers had compromised as well. That other website in turn ran the LightsOut exploit kit, which executed exploits against a visitor's browser or browser plugins to install the malware.

In both cases, once inside the networks of the power company, the malware executed ICS specific code to search for and later access target ICS servers. Lastly, the attackers compromised the websites of three different ICS equipment providers, and subsequently trojanized the software bundles that were available for download. The installation of those software bundles can bring ICS malware directly onto the human-machine interface (HMI) machines, i.e. computers, engineering machines, and even on protective relays.

Stuxnet was able to propagate over the network by exploiting a vulnerability in the Print Spooler service, and a vulnerability in the Server service [5]. The reader is referred to the Microsoft security bulletins MS10-061 and MS08-067, respectively, for a description of the root cause of those vulnerabilities. Stuxnet was also able to propagate through network shares. The ICS worm may travel a shorter range when the target industrial facility is predetermined and located ahead of time. Stuxnet could land on the networks of a target industrial facility right away if an insider purposely plugged a Stuxnet-infected removable flash drive into a Windows machine in those networks. Stuxnet modified the Siemens ICS engineering software SIMATIC WinCC/Step 7, more specifically the dynamic-link library `s7otbxdx.dll` of SIMATIC WinCC/Step 7, to inject attack MC7 code into target programmable logic controllers (PLCs).

MC7 is the compiled assembly of code written in the Structured Text (ST) programming language. ST in turn is one of the five programming languages supported by the IEC 61131-3 standard to program PLCs. The MC7 code injection was done over the network via the S7comm protocol, i.e., a Siemens proprietary protocol used for PLC programming, data exchange between PLCs, and PLC data access from supervisory control and data acquisition (SCADA) systems [6]. Stuxnet turned a compromised machine into a modified S7comm client and was mostly selective when searching for its targets, although unintended machines were compromised during the attacks.

Although the ICS attack code may be inside the networks of an industrial facility, there is still way to go. A machine that the ICS attack code has just compromised may not be usable to attack or manipulate field devices. Stuxnet, for example, needed to compromise the so-called SIMATIC Field PGs, which are Windows machines used to program PLCs. Stuxnet spread within the networks of a target industrial facility. It hopped from machine to machine until landing on a SIMATIC Field PG, from whence it injected attack code into PLCs and hence sabotaged the physical processes they were controlling. Another ICS attack code, namely IronGate, uses attack techniques that are similar to those of Stuxnet [7]. IronGate operates at medium range, and searches for very specific target machines, as Stuxnet did. Nevertheless, IronGate was deemed to be simply a proof of concept or research activity, given that it is not associated with any attack campaigns or threat actors [7].

Not all interactions of ICS malware with power grid equipment are actual attacks aiming at causing physical damage right away. As in the case of Dragonfly, ICS malware may limit their operations to spying on the power grid, which entails the following operations:

- Identifying power equipment and substation physical processes along with their parameters. This operation aims at inferring the substation layout, which can later be reconstructed as diagrams with power lines, busbars, switches, circuit breakers, and power transformers.
- Locating and characterizing field devices, i.e., protective relays, aka intelligent electronic devices (IEDs), PLCs, and phasor measurement units (PMUs). This operation discovers their system and network configurations, including their IP addresses. It also determines their substation functions, such as transformer protection, feeder protection, etc., and hence discovers the cyber-physical mappings. A cyber-physical mapping indicates which field device monitors and controls what specific substation equipment and physical process.
- Locating and characterizing HMI machines, engineering machines, routers, and possible firewalls and anti-malware systems currently in use.

Target selection In ICS malware, it can be highly cognizant of the mechanics and physics of physical equipment and processes in an electrical substation. Those details are quite visible to a knowledgeable attacker through sophisticated analysis of network traffic and control system activity, i.e., machine and IEC 61131-3 code, I/O traffic, operating system events, etc. That is why traditional information technology (IT) honeypots are unable to trap a knowledgeable attacker in an industrial facility.

Industrial honeypots One of the most important developments in cyber deception is the honeypot concept. There are a number of works in this field, including variants for industrial process control networks. One notable example is a PLC honeypot by Buza et al. known as CryPLH [8]. CryPLH implements several PLC services that are integrated into a Linux-based virtual machine (VM), which forms the honeypot. Another PLC honeypot is CONPOT, which was designed and implemented by Rist et al. [9]. CONPOT relies on Python scripts to emulate a range of common industrial communication protocols. Yet another industrial honeypot was developed by Vollmer and Manic [10]. Their industrial honeypot is able to do self-configuration based on passive observations of control system network traffic.

CryPLH, CONPOT, and other industrial honeypots along their line of work, focus on presenting a convincing PLC interface once an attacker has actively connected to the honeypot. They do not, however, provide a means to get the attacker to pursue the honeypot in the first place, nor can they sustain interactions with the attacker after an initial exchange of network packets. By comparison, our work seeks to create a more convincing target by simulating a greater feature set, which can sustain long term communications with the attacker.

CryPLH, CONPOT, and other industrial honeypots are characterized by an absence of system and network activity. By contrast, however, industrial control systems such as protective relays in electrical substations are constantly in action. They read from sensors, analyze data, submit reports to human operators, and send commands to actuators. Industrial process control is a highly dynamic operation full of system and network activities. Industrial honeypots cannot operate on industrial control systems in production, consequently need dedicated computers to run on. Furthermore, the substation data that are stored on the flash drives of industrial control systems, or that are transmitted over the network, obey the physics laws of the diverse physical processes and equipment operations that take place in an electrical substation. Industrial honeypots do not have the capability to reproduce and expose control system dynamics for the purpose of defensive deception against ICS malware.

3 The Prospect of Industrial Mirage

Substation equipment reflect in cyber space All physical equipment of an electrical substation, and the physics therein, are reflected in industrial control systems and networks, as well as in general-purpose computers involved in the monitoring and control of the electrical power grid. They are all visible on those systems, which in this chapter are collectively referred to as ICS machines. The reflection is manifested in various forms, depending on the range from which those ICS machines sense and operate an electrical substation over the network. At the level of protective relays, i.e. industrial control systems that are directly attached to substation equipment, with the advent of IEC 61850 standard [11], the reflection is manifested explicitly as data objects called logical nodes that model substation equipment and processes.

At the level of HMI computers, the reflection in question is manifested in a highly distributed form. It mainly consists of I/O data stored in various buffers, as well as substation-specific computing and network traffic. A similar manifestation is observed at the level of SCADA machines. These are computers that enable a human operator to monitor and operate an electrical substation from a remote location over a long distance network.

Industrial mirage The concept refers to a reflection of substation equipment and related processes that do not exist, but appear as real in cyber space at various ranges relative to the site where they are hosted. They are in fact virtual substation equipment and emulated physics with no real counterparts. In our prior work [12], we explored analog-to-digital (A/D) and D/A conversion as a physical barrier to hide the virtual equipment and emulated physics from malware. However, the approach did not scale well and had limited deployability.

Research question The ultimate objective of industrial mirage is to protect the electrical power grid from ICS malware developed by knowledgeable attackers. Our research investigates on how to develop an effective industrial mirage, and most importantly how to leverage an industrial mirage to attain the following security and usability goals:

- Detect malware bound for the electrical power grid without any prior knowledge of their code and data. Such zero-knowledge detection needs to be performed on ICS machines that are in actual production.
- Enable cyber operations against attackers for attribution purposes.
- Enable an ICS machine to isolate the malware and fully recover from them.
- Have 0-interference in the legitimate work of ICS machines, and an absolute safety towards the electrical power grid.

Composition Industrial mirage consists of a stack of deception layers as summarized in Fig. 1, which are integrated into the software, firmware, and hardware of an ICS machine. They all work together to project an industrial mirage onto cyber space, and leverage it to redirect the target selection of ICS malware towards decoy substation equipment. Particularly useful is operating system (OS) kernel coding that engineers decoy targets and target search mechanisms with foundations in decoy I/O devices such as network interface cards (NICs), disks and solid state drives, and I/O boards. Another important layer in the stack consists of deception algorithms, which are based on mathematical learning of adversarial cyber interactions and are supported by decoy I/O devices in the OS kernel.

The zero-knowledge detection of ICS malware performed by industrial mirage includes a second-order form. Second-order detection refers to detection at a time that postdates the data interception mounted by ICS malware. The deception algorithms covertly inject various decoy data into the data stream of ICS malware in order to affect their behavior and overall operations. This enables industrial mirage to use deception-guided big data analytics to attain second-order detection of ICS malware. Industrial mirage aims at countering all types of malware bound for the

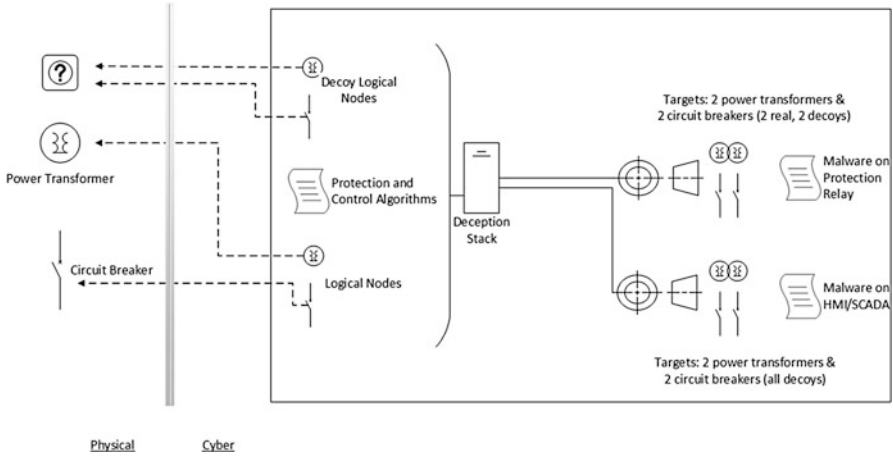


Fig. 1 A deception stack projecting industrial mirages of substation equipment for ICS malware to pursue

electrical power grid, including kernel-level malware, which by definition have direct access to the underlying hardware. Consequently, industrial mirage relies on hardware support provided by a deception processor.

4 OS Kernel Interventions

4.1 Decoy I/O Devices

Creating imaginary networks A decoy network interface card (NIC) has shown to be an enabler of industrial mirage. We designed and implemented a decoy NIC by researching kernel driver techniques for Windows [13]. The central component of a decoy NIC is a low-level deceptive driver, which is integrated with other drivers specialized in handling real NICs. The concept is depicted on the left lower part of Fig. 2. In Windows, the drivers that manage an I/O device in general are organized in a stack. As applications issue system calls to send and receive data, a component of the Windows kernel called I/O manager packages the request data into an I/O request packet, which it sends to the driver at the top of the stack. Once that driver is done with its processing of the I/O request packet, it passes the packet down to the lower driver. Eventually the I/O request packet reaches the lowest driver, i.e. the deception driver in our case.

Instead of interacting with a real NIC controller through the main CPU or a direct-memory-access controller, the deception driver emulates the underlying hardware and all protocol interactions with that hardware. I/O device emulation at the device-driver level is advantageous to the defender, given that by definition the I/O

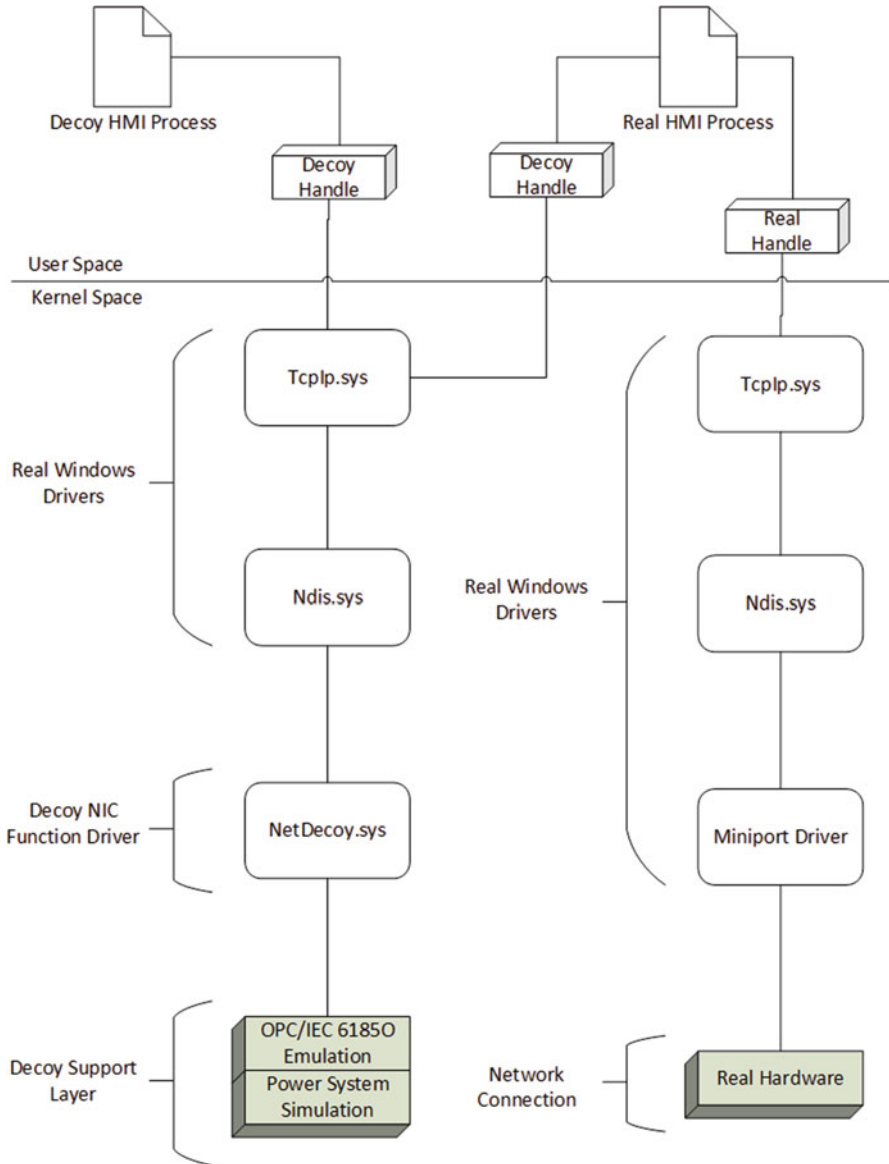


Fig. 2 Decoy NIC along with decoy and real HMI process handles

subsystem of the OS kernel is not supposed to be aware of the underlying hardware and protocol communications. The deception driver is linked with Windows kernel libraries, which enable it to interact with the upper drivers in the stack identically to how a real driver does. Now the ICS malware see an imaginary network that they think they may use to reach and exploit ICS machines. The decoy NIC is consistent and resembles closely its real counterpart depicted on the right lower part of Fig. 2.

Auxiliary decoy I/O devices We use similar techniques to realize decoy secondary storage, i.e. disks and solid state drives. Decoy secondary storage is used to store decoy HMI applications, their configuration files, and decoy substation data. Our goal is to create an attack surface over which to leak decoy data. Some of the decoy data come from decoy secondary storage, while others come from a decoy keyboard. Similarly to a decoy NIC, a decoy keyboard is based on a low-level deception driver, which emulates and exposes keystrokes modeled after actual users. The decoy keyboard is able to shadow the physical keyboard so that one single keyboard appears on the machine at all times. When the deception driver detects a time window of user inactivity, it detaches the physical keyboard from the device tree and attaches the decoy keyboard there. When a user returns to work and presses any key on the physical keyboard, the deception driver receives the keystroke and hence immediately restores the physical keyboard in the device tree [14]. A decoy mouse is realized using the decoy keyboard's algorithms as well.

Decoy I/O boards Protective relays use I/O boards to sense and actuate substation equipment. Sensors provide measurements through the I/O control inputs of an I/O board, for example phase current measurements, for a protective relay to read. A protective relay can send commands, for example trip a circuit breaker, by writing to the I/O control outputs of an I/O board. We developed a decoy I/O board to make a protective relay appear to be connected to substation equipment, which are in fact industrial mirages. A decoy I/O board marks the end of the decoy I/O line. We realize a decoy I/O board by emulating an I/O board controller in the OS kernel. This is a layer of code that mimics a special-purpose microprocessor that operates and manages an I/O board, which is a decoy as well. As in the case of a decoy NIC, a deception driver drives emulation rather than actual hardware. The end result is an industrial mirage that can be sensed and actuated locally on the protective relay.

4.2 *Decoy User-Space Activity*

Decoy processes The creation of an attack surface for ICS malware to pursue is aided by decoy OS processes, which appear to work with decoy I/O devices. With reference to our example, there are decoy processes to mimic an HMI application that sends and receives data over a decoy NIC, as well as read and write files that reside on decoy secondary storage. The introduction of decoy processes in the equation is depicted at the top of Fig. 2. Some of the decoy processes appear to run without interruption, while others, such as a decoy controller configuration tool in memory, appear to run only occasionally. The decoy user activity is realized by instrumenting OS data structures that pertain to processes and their threads, as well as techniques to maintain safety. For example, it is important that the central processing unit (CPU) scheduler does not dispatch the CPU to decoy processes and hence waste CPU cycles.

Deception goals The desirable effect of decoy processes is twofold, namely to make any Windows machine, whether it is ICS related or unrelated, appear as a

real-time automation controller or SCADA client, and to make a decoy NIC on that machine appear as sending and receiving substation network traffic. That effect is intended to serve the goal of causing ICS malware to conclude that a compromised machine is a valid target; that the target is directly or indirectly connected to an electrical substation; and that a decoy NIC leads to other machines in the electrical substation, including field devices and data concentrator machines. As defenders, we want ICS malware to exercise ICS specific search and attack techniques over a decoy NIC. We know from our previous work that Windows non-ICS goodware may send network traffic over a decoy NIC, however the fact that the traffic in question is not part of a valid ICS protocol communication can be validated reliably and in little time [15]. Because we take measures to ensure that a human does not send network traffic over a decoy NIC, incoming valid ICS communications unequivocally expose the source as ICS malware.

With reference to Fig. 2, the decoy processes mimic an HMI application running on a real-time automation controller. An automation controller machine in a real electrical substation already runs real HMI application code, consequently it leaves no room for decoy HMI processes. We do not run decoy HMI processes on a real-time automation controller. We only display a decoy presence of network communications between those real HMI processes and a decoy NIC. We do so to redirect at least some of the network operations of ICS malware onto a decoy NIC. Even limited malicious network communications over a decoy NIC would suffice to detect the ICS malware and hence identify their location in memory. This explains the presence of decoy handles that are made to appear as if they existed within the address space of a real HMI process.

Data instrumentation The main technique that we use to display decoy processes is to instrument the OS data structures that pertain to processes and their threads. We instrument data structures both in user space and in kernel space. A diagram of the data structure instrumentations is depicted in Fig. 3. In user space, a decoy process is supported by a process environment block (PEB), which is identical to the PEB of the process' real counterpart. For example, the decoy PEB contains a pointer to a doubly-linked list of loaded modules, as well as a session identifier, a path of the image executable file, and input arguments. A decoy process is also supported by thread environment blocks (TEBs), which display a state for each decoy thread. A decoy process is given a virtual address space, which is mapped to a file on the decoy secondary storage and hence does not consume any physical main memory frames.

Other user-space support for a decoy process includes an entry in a linked list maintained by the Client/Server Runtime Subsystem process, or csrss.exe, which in turn is considered to be the user space component of the Windows subsystem. The entry at hand describes the structure of a decoy process and its threads, just like the other entries describe real processes and their threads. A decoy process has a representation in the kernel component of the Windows subsystem as well. More specifically, the Windows subsystem maintains a doubly linked list in the kernel, where each process is reflected in a data structure. We populate and add one for a

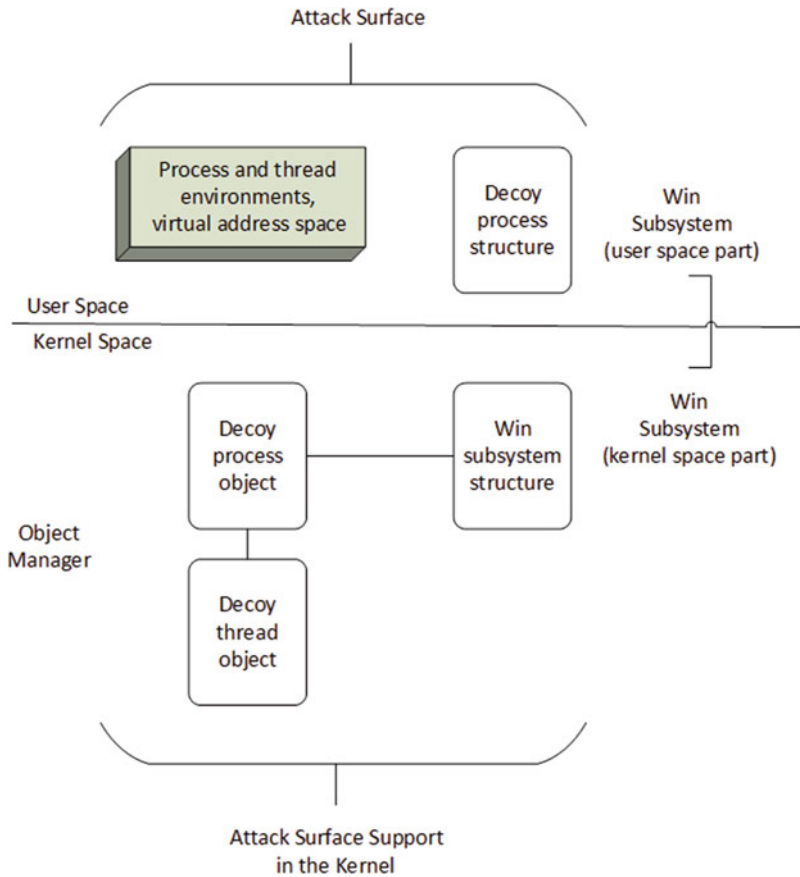


Fig. 3 Data structure instrumentation that displays a decoy process

decoy process. The most potent data structure that models and represents a process is the EPROCESS. Similarly, a thread is modeled by an ETHREAD. We develop an EPROCESS for a decoy process, as well as an ETHREAD for each of its threads. These data structures are encapsulated as objects by the Object Manager of the Windows kernel.

Discoverability The data structure instrumentations make the decoy processes discoverable. For instance, the `ps` command lists the names of the decoy processes along with the most important parameters that characterize them. The `process viewer` tool from the Sysinternals suite finds and details the decoy processes as well. Also, invoking application programming interfaces (APIs) of the process status API (PSAPI) library results in the discovery and detailing of the decoy processes. The discoverability of decoy processes creates a decoy attack surface, which ICS malware could examine, pursue, or do both. ICS malware could mount local exploits

against a decoy process. ICS malware could land on its target machine through removable media, or over one of the real NICs. As it is shown in Fig. 3, a decoy process is paired with a decoy NIC and hence an imaginary network that leads to decoy targets, i.e. emulated field devices and other substation machines.

4.3 Decoy ICS Targets

Decoy I/O devices are a pivot for target emulation The modus operandi of ICS malware involves a search for targets, which is performed by querying network resources [Dragonfly and BlackEnergy], or by analyzing code and data on the file system of the compromised machine [Stuxnet and IronGate]. Decoy I/O devices are leveraged to interfere with these target searches. For example, a decoy NIC can display an imaginary network along with decoy Object Linking and Embedding (OLE) for Process Control (OPC) protocol servers [16]. The decoy OPC servers appear to be reachable over the imaginary network. If ICS malware want them, they need to access the decoy NIC. This principle forms the basis of zero-knowledge detection in industrial mirage. ICS malware are detected on the first encounter solely on the basis of their interaction with a decoy I/O device and the decoy targets that it leads to [15].

Exploring ICS mechanics Now that an imaginary carrier is displayed for ICS malware to see, i.e. an imaginary network in the example of Fig. 2, a decoy target needs to be realized. We developed target emulation engines for that purpose. The decoy support layer shown in the lower left corner of Fig. 2 includes target emulation engines that we designed to work closely with the deception drivers discussed previously in this chapter. They use the mechanics of ICS protocols, HMI applications, SCADA tools, protection and control algorithms, and control system configuration tools, to realize decoy targets that resemble realistically their real counterparts. The decoy targets can be discovered and attacked by ICS malware without a single network packet ever leaving the boundaries of the OS kernel of the compromised machine.

For example, the deceptive emulation of IEC 61850 referenced in Fig. 2 maintains logical devices, each of which is a composition of logical nodes that have common features. Those logical devices together form a physical device that represents a protective relay, which in reality does not exist but yet appears to be existent and reachable over a decoy NIC. The deceptive emulation includes network communication services defined as per the IEC 61850 Abstract Communication Service Interface (ACSI). Some of the services offer a client-server model, which ICS malware can use to set or get substation data. Other services offer a peer-to-peer model, in which ICS malware may be involved to maliciously exchange data with its target protective relay. Either form of interaction with a decoy protective relay establishes a path for transmission of data over the network, i.e. an association in the IEC 61850 language, which leads to immediate detection.

Given that IEC 61850 is a virtual protocol, it needs an actual carrier. The deceptive emulation maps the IEC 61850 data and services to the Manufacturing Messaging Specification (MMS) [17]. It is very common in actual electrical substations for IEC 61850 to be mapped to a protocol stack comprised of MMS, TCP/IP, and Ethernet. MMS was designed specifically for transferring in real time large volumes of physical process data and supervisory commands. MMS has object models and services that can easily accommodate those of the IEC 61850 standard. The physical process data that the deceptive emulation serves to ICS malware are decoys as well. For consistency reasons, those data are generated by using existing power system simulation tools such as the real time digital power simulator (RTDS) [18]. Their simulation algorithms ensure that the decoy data comply with the physics of an electrical substation. Those simulation algorithms per se are not part of industrial mirage, but are instead used by industrial mirage as they are.

4.4 Safety and Usability

The main challenge stems from humans, i.e. system operators, power engineers, device technicians, or even simple users, accessing decoy I/O devices, decoy targets of any kind, and decoy OS processes and other consistency mechanisms that the deception algorithms may engage on the computer or ICS machine. For example, a system operator could run by mistake a decoy HMI application stored on a decoy disk partition. The decoy HMI application could connect to a decoy OPC server over a decoy NIC, and retrieve the decoy substation data stored in decoy OPC objects. In addition to raising numerous false positives, the system operator could take action on an electrical substation based on the decoy data. Equally dangerous is a situation in which a power engineer writes protection and control code that acts on the I/O control inputs and outputs of a decoy I/O board.

Figure 4 illustrates the research path that we have found to show potential to make the mirage-centric capability absolutely transparent to people with legitimate access to computers and ICS machines. Search-and-filter techniques, which operate within the driver stack of a monitor, analyze and possibly revise picture frames bound for the monitor. Those techniques are designed to remove all entries that pertain to decoy I/O. The removal takes place before those entries are visualized on a monitor along with other data for a human to see. With reference to Fig. 4, the IP addresses of two decoy OPC servers and the data of two decoy IEC 61850 logical nodes, namely a decoy circuit breaker (XCBR) and a decoy power transformer (YPTR), are removed on the fly. If a system operator does not see the decoy OPC servers, he/she will not be able to connect to them. Similarly, if the system operator does not see the decoy XCBR and YPTR, he/she cannot trip the decoy circuit breaker, nor can the operator take action based on any data that pertain to the decoy power transformer.

We have experimented with some basic keyword-based search-and-filter techniques in an OPC environment, which we discuss in detail in [15].

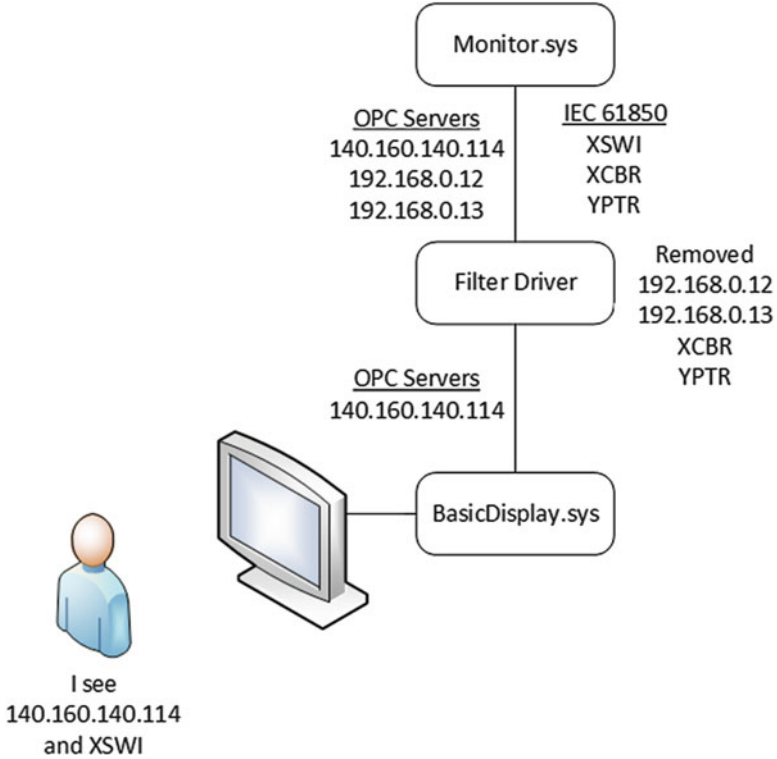


Fig. 4 High level illustration of the monitor filtering capability

5 Other Tools and Techniques

The elements of industrial mirage, except the deception-guided big data algorithms, run on ICS machines that reside within an electrical substation, namely HMI machines, i.e., real-time automation controllers, engineering servers, and protective relays. They also run on machines that reside in the enterprise networks of a power utility company, as well as engineering laptop machines. These machines perform power grid related functions. Some of them may have remote virtual private network (VPN) access to substation networks. Industrial mirage may also run on other machines on the Internet that are not related to the electrical power grid. In this latter case, the defense is particularly useful against ICS malware that have a worm-like propagation. As to the deception-guided big data algorithms, they run on a big data computing platform of general use.

5.1 Deception Algorithms

Purpose The deception algorithms drive mirage operations such as to maximize the likelihood of decoy I/O devices and emulated targets being pursued by ICS malware. The deception algorithms choose optimal consistency measures, depending on the current dynamics of the system. They determine the deceptive environment that is presented at any point in time. After ICS malware are detected, the deception algorithms sustain continuous interaction with attackers in support of cyber operations against them. An unconventional goal is to exploit cyber techniques such as to push adversarial operations onto the physical world, where counter intelligence personnel can use traditional human intelligence (HUMINT) methods to attain attribution. Other goals pertain to generating wanted data that are particularly useful for attack attribution.

Mathematical modeling Nonlinear dynamics [19] has shown potential to characterize the decision making of ICS malware, which is modeled as a complex system composed of nonlinear constituents. Of particular interest are the emergent patterns of the behavior of such complex system. These patterns of behavior originate from the low-level constituents of the complex system in question, but are not observable in its constituents individually. The main research challenges here consist of finding a correct mathematical formulation of the complex system, determining emergent patterns in the behavior of the complex system, and how to actively cause the appearance of those emergent patterns of behavior. Mathematically, we view adversarial behavior as specific temporal evolutions of the states of the nonlinear dynamical system in the associated phase space.

Adversarial behavior is characterized as mathematical chaos. This means that the system-level events and network packets that are generated by ICS malware originate from a set of deterministic local rules that govern the ICS malware's decision making. Although those system-level events and network packets are generated deterministically, they appear to be random and unpredictable due to the effect of conditions in the computing environment. Even a small change in those conditions can alter to a large degree the decisions made by the malware. Such randomness makes the identification of emergent patterns of adversarial behavior impossible by traditional statistical-norm data mining. This characterization of adversarial behavior may mean that the temporal evolutions of the states can be defined by a system of nonlinear difference equations. The trajectories of those evolutions appear as irregular or random motion in phase space as the system has sensitive dependencies on initial conditions [20].

Controlling ICS malware's behavior mathematically Industrial mirage uses chaos control theory [21, 22] for mathematical control of adversarial behavior. The assumption in most cases is that a target adversary, i.e. ICS malware or the people behind them, would respond to probes only if the information conveyed in the probes is of interest. At least in signal systems in biology, receivers of signals

respond only if that benefits them, as a synthesis of reliability and deception in those signal systems suggests [23]. The adversarial models can evolve using genetic algorithms [24] to avoid an overly local focus in optimization. With regards to sustaining prolonged interaction with attackers, nonlinear time-series analysis is used to reconstruct the underlying chaos of adversarial behavior from captured malware data.

Critical tasks include identifying the state or condition variables that define the states of the nonlinear dynamical system, i.e. ICS malware's decision making, and the input or control variables to which the nonlinear dynamical system is highly sensitive. Of particular importance are also the systems of nonlinear difference equations that represent the local chaos-generation rules of the nonlinear dynamical system at hand. Knowledge of the chaos-generation rules may help with the engineering of a more accurate redirection intervention on ICS malware. The deception algorithms include elements of estimation and control based on control theory [25]. The motivation for control theory is to be able to run deception algorithms autonomously, and also be able to conduct self-management and tuning. State estimation techniques that have shown potential for coping with chaos noise in ICS malware behavior are the nonlinear or unscented Kalman filters [26].

Interaction with other I/O devices Deception algorithms work also with decoy keyboards, decoy mice, and decoy webcams, to enable second-order detection and to better guide cyber operations. These are devices that are intercepted by ICS malware as well, although the ultimate target is the electrical power grid. Deception algorithms are stochastic in nature and hence unpredictable.

Covert communications With malware penetrating the OS kernel, the computer becomes a battlefield where ICS malware code and industrial mirage code operate in close range to each other. Industrial mirage uses steganographic methods [27] to enable deception algorithms, decoy I/O devices, decoy targets, consistency measures, and the deception processor, to communicate with each other secretly over channels that are possibly interceptable by malware.

5.2 *Deception-Guided Big Data Analytics*

The need for big data analytics Malware may access decoy I/O devices and decoy targets when under the influence of decoy data, in which case a second-order detection is attained. For example, a deception algorithm may use a decoy keyboard to leak fake credentials that provide access to a VPN, which appears to be reachable only over a decoy NIC. VPNs in the business network of a power utility company are of particular interest to attackers, since some of them provide remote access to electrical substation networks. The use of the fake credentials over the decoy NIC leads to an unequivocal second-order detection of the ICS malware. However, ICS malware may access real I/O devices too as a result of decoy data. For example, malware may trip a real circuit breaker with the intention of disrupting a decoy

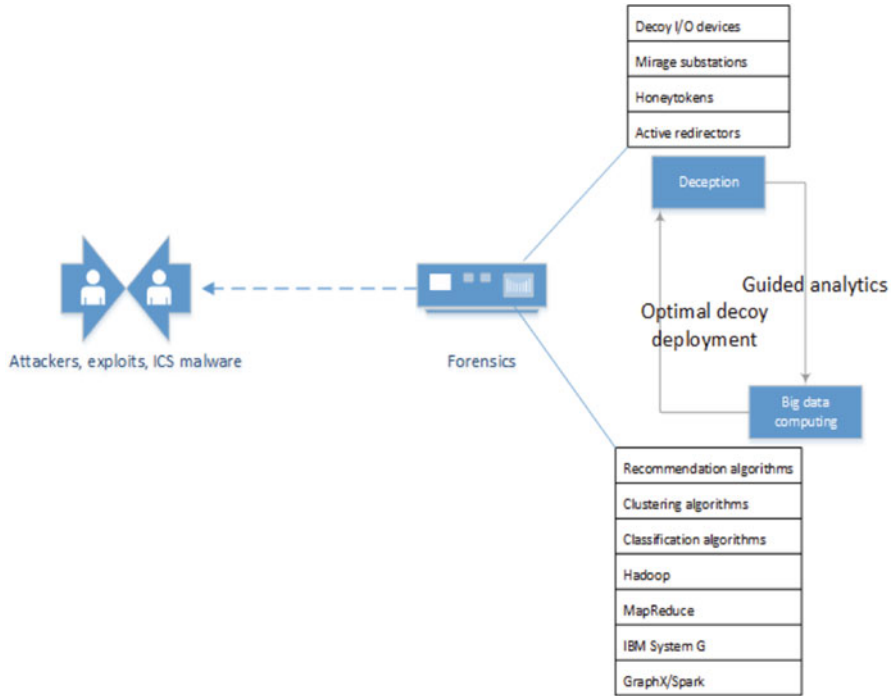


Fig. 5 Big data computing integrated with industrial mirage

power transformer. In this latter case, second-order detection can be attained only by collecting and analyzing data from ICS machines other than decoys. Industrial mirage uses big data analytics that is guided by highly specific knowledge coming from its various decoy mechanisms. The concept is illustrated in Fig. 5.

Same platforms, but new algorithms The platforms for big data computing that we use in industrial mirage are those of big data computing in other fields where big data analytics is utilized. More specifically, Hadoop [28] and MapReduce [29] work very well on running deception-guided big data algorithms. However, the algorithms need to be specialized and thus are to be designed specifically for that purpose. Merely adapting existing big data analytics algorithms from other fields does not work. Recommendation algorithms in various forms, namely content-based filtering, collaborative filtering, and hybrid filtering, are particularly useful with regards to predicting malware operations based on knowledge of decoy data. Clustering algorithm can leverage relations between decoy data to group individual pieces of data and/or code that belong to malware. Classification algorithms can predict how malware interact with real I/O devices based on past observations of their interactions with decoy I/O devices. Of practical value are also distributed algorithms over very large graphs and matrices, which have shown potential to model and recognize complex interactions between real data and decoys.

A two-way relation So far we discussed how decoys can guide big data algorithms for zero-knowledge malware detection, but their cooperation goes both ways. Big data algorithms can help with determining what specific decoy data and other decoy mechanisms to use, and where specifically in the electrical power grid to deploy them.

5.3 Future Work: Hardware Support

Rationale Kernel-level ICS malware are in the position to initiate reconnaissance probes directly on hardware buses for the purpose of verifying the existence of their target I/O devices. Given that decoy I/O devices do not exist, there would be no replies returned. If not addressed, this issue would result in a complete loss of the defense value of decoy I/O devices along with the decoy targets that they support. Furthermore, hardware support has potential to help with all the mirage tasks discussed previously in this chapter.

Research direction We are researching solutions based on a programmable gate arrays (FPGA) device, which will be added to the overall decoy system architecture at the hardware level. This deception processor will respond on hardware buses on behalf of all decoy I/O devices projected on the computer or ICS machine, and will also act like a device controller for them. The deception processor will be attached to the main circuit board, and will have access to the main peripheral component interconnect (PCI) bus, as well as to the bus that connects the primary microprocessor with the physical memory. Research on hardware solutions is not unprecedented. There is already an established line of related work that explores hardware support solutions for security. Examples range from making a software system to be copy and tamper resistant [30], to certifying program execution [31].

Protection of own code and data This is a challenge that affects not only industrial mirage, but also all anti-malware systems, academic and commercial. ICS malware that penetrate the OS kernel are expected to attack the security mechanisms so that to disable them or otherwise bypass the defense that they provide. Industrial mirage includes algorithmic uses of memory manager components to trace accesses to memory regions of interest. As with decoy I/O devices, discovery probes issued by malware can be used to detect those malware. Of interest are address translation and page fault handling along with memory mapping techniques, combined with hardware support, which can be leveraged to hide all code and data of industrial mirage, without losing the ability to run the deception capability. Decoy I/O devices expose no attack surface over the network, since their input comes only from within the machine on which they are deployed.

6 Evaluation

Realistic testbed We tested industrial mirage in a research testbed that resembled a substation network. The research testbed was comprised of real-world ICS machines, which are instances of industrial control systems that are widely deployed in the electrical power grid in North America. All machines in the research testbed were connected on a local area network. There were no connections to any outside networks due to safety reasons. A list of the ICS machines in our research testbed and their main characteristics are given in Table 1. The SEL-487E-3 is a protective relay that can monitor and protect a power transformer from electrical faults. It runs intelligent algorithms to detect various types of faults. It is able to take action in a timely manner by operating electrical circuit breakers and disconnect switches. The SEL-421-4 is a protective relay that can perform industrial automation functions. It includes 32 programmable elements for local control, remote control, automation latching, and protection latching.

A SEL-421-4 can also perform various functions to protect overhead electrical transmission lines and underground cables. The SEL-3355 performs several substation functions as well. It has an integrated HMI, with a local display port. In this line of work, the SEL-3355 operated as a real-time automation controller. The SEL-3355 periodically polls the SEL-487E-3 and SEL-421-4 protection relays to collect substation data from them. The S7-1500 is a PLC from Siemens. It has I/O modules that can be directly attached to sensors and physical equipment. The network communications between the SEL machines take place over the distributed network protocol (DNP3) [32] and IEC 61850. We integrated an engineering server into the research testbed. The engineering server hosted an OPC server, which in turn ran an IEC 61850 protocol driver to get substation data from the SEL machines.

One of the general-purpose Windows machines acted as a field programmer (PG) towards the S7-1500. It ran the totally integrated automation (TIA) portal, and hence included ICS software such as Step7 and WinCC. The field PG machine communicated with S7-1500 over the S7comm protocol.

Remote data acquisition attacks We tested industrial mirage against a large number of OPC malware samples involved in the Dragonfly cyber espionage

Table 1 Testbed machines and their main ICS protocols

Machine	ICS function	ICS protocol
SEL-3555	Automation controller	DNP3, IEC 61850
SEL-487E-3	Transformer protection relay	DNP3, IEC 61850
SEL-421-4	Protection, automation, and control	DNP3, IEC 61850
Windows	OPC server, PG	DNP3, OPC, IEC 61850
Windows	OPC client	OPC
Windows	None	None
SIMATIC S7-1500	Programmable logic controller	S7comm

campaign. There are many versions of those malware samples, all of which are publicly available for research on academic malware repositories. We ran the ICS malware samples on the OPC client machine and on the other Windows machine with no apparent power grid functions. All of these malware samples were intercepted by industrial mirage when they attempted to discover target OPC servers over an imaginary network created by a decoy NIC. We also ran rogue IEC 61850 client tools and S7comm tools on all Windows machines of the research testbed. Industrial mirage intercepted those rogue clients when they tried to use the MMS and S7comm protocols to connect to decoy IEC 61850 and S7comm devices, respectively, on the imaginary network. A sample of data seen by the decoy NIC while those network attacks were taking place is given in Fig. 6.

Some of the OPC data items on the real OPC server were paired with attributes of decoy IEC 61850 logical nodes on the protective relays. The decoy IEC 61850 logical nodes, in turn, were mapped to I/O control inputs of a phantom I/O board. In these circumstances, we let all malware samples discover the engineering server. They connected to the OPC server, and collected over OPC the substation data originating in the protective relays over IEC 61850. Furthermore, we directed the rogue IEC 61850 client tools to retrieve substation data directly from the protective relays. Some of those data were the ones to be mapped to I/O control inputs of a phantom I/O board. Industrial mirage intercepted all these attacks when the decoy I/O board controller received requests to read the I/O control inputs of the phantom I/O board. A sample of data that were seen by the decoy I/O board controller as these attacks were taking place is plotted in Fig. 7.

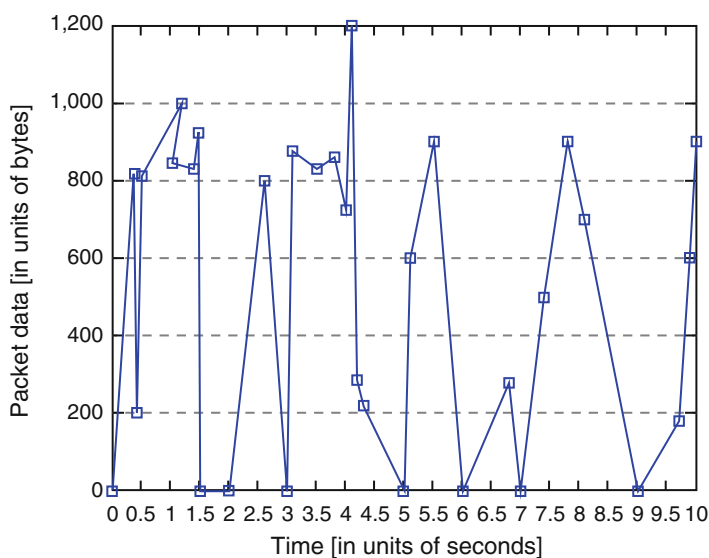


Fig. 6 Packet data arrivals on the decoy NIC (Combined malicious OPC, IEC 61850/MMS, and S7comm traffic)

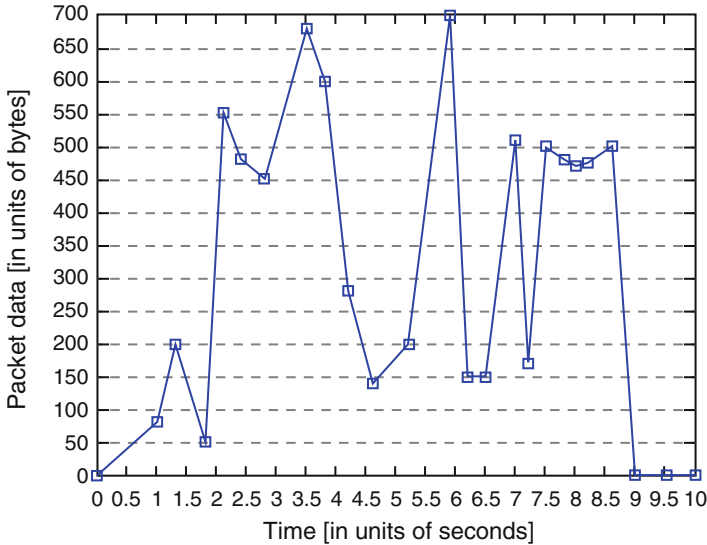


Fig. 7 Data traversing the decoy I/O board controller (Combined OPC and IEC 61850/MMS attacks)

When no malware are running, there should not be any data traversing decoy NICs or decoy I/O board controllers, unless there is a failure of the safety and usability tools. An experimental feature of decoy I/O consists of using decoy I/O devices to generate synthetic data, which in turn result in traffic of our own traveling over the driver stack of decoy NICs or decoy I/O board controllers. Since our code is the source of the synthetic data in question, we can easily differentiate own traffic from any other traffic. The transmission of own traffic over the driver stack of decoy I/O devices in general serves the purpose of making those devices appear to be in full operation.

Remote command injection attacks This part of the testing involved BlackEnergy-style attacks. BlackEnergy had a keylogger module that intercepted keystrokes on compromised machines in the enterprise network of a target utility company. The keylogger intercepted VPN credentials and subsequently used those credentials to access substation networks over a VPN. At that point, the attackers ran rogue SCADA client software on the compromised machines and hence sent remote commands to an electrical substation [2]. We replayed the BlackEnergy attacks in order to send remote actuator commands over the network to the protective relays in our research testbed. The purpose of the actuator commands was to trip circuit breakers. Although BlackEnergy operates on substation equipment similarly to legitimate system operators, there is a key difference between them.

Because of the monitor filtering capability, a legitimate system operator does not see decoy NICs, decoy I/O boards, and any form of industrial mirage, unlike an attacker who sees them all as real. We used a decoy keyboard as discussed earlier

in this chapter to leak VPN credentials, which in turn redirected the attacks onto a decoy target over the decoy NIC. The attacks were detected when the malicious network traffic landed on the decoy NIC. When the rogue SCADA client software communicated over a real NIC, the attacks were detected too. Some of the circuit breakers tripped by the remote actuator commands were industrial mirages. The actuating requests reached the decoy I/O board controller, asking to access I/O control outputs of the phantom I/O board, which is when detection was attained. The data encountered by the decoy NIC and decoy I/O board controller are somewhat similar to those plotted in Figs. 6 and 7, respectively. The only difference is that the requests are fewer, and the data sizes encountered are generally much smaller.

Local sensing and actuating attacks This part of the testing involved the emulation of ICS malware attacks on protective relays. The test code obtained direct readings of sensors and also injected actuator commands bound for circuit breakers. These attacks were detected because the I/O operations reached the decoy I/O board controller. The test code also modified an actuating command on the fly shortly after it was issued by a decoy protection and control application on the protection relay. This other attack was detected because of a mismatch between the command that arrived at the decoy I/O board controller and the command that the decoy protection and control application was driven to issue. Yet in another attack, the test code captured an actuating command and discarded it entirely. This other attack was detected because the decoy I/O board controller was expecting the actuating command and its waiting period expired. Same results were obtained when the actuating command originated in the decoy NIC before being modified on the fly or entirely suppressed by the test code. Same results were also obtained when sensor measurements were manipulated or denied by the test code.

No false positives Because legitimate system operators cannot see the decoy I/O devices, industrial mirages, and decoy entries in general along with their respective data, on their monitors, they cannot act or react on any of them. On the other hand, the decoy user-space activity that was discussed earlier in this chapter makes it appear as if they do access decoy I/O, take action upon decoy data, and monitor and control industrial mirages as well. This makes decoy I/O and industrial mirages in general quite indiscernible from their real counterparts, while keeping the approach completely transparent to system operators. Consequently, because the various traps do not receive any normal activity from system operators, no false positives have been observed.

Execution overhead We express the measure of overhead as the approximate average fraction of 1 millisecond that goes to executing instructions of decoy I/O. Although this measure is an estimate, it clearly quantifies the effort that the CPU makes to run decoy I/O instructions. We work with the million of instructions per second (MIPS) metric of the CPU on the machine on which decoy I/O is deployed. MIPS metrics are measured by Dhrystone computing benchmark programs under realistic workloads. Dhrystone's measurements are cognizant of instruction pipelining. We acquire high-resolution time stamps at various points in

the code. These points are organized as `StartingTime` and `EndingTime`. We subtract `StartingTime` from `EndingTime` to obtain the length of the time interval for which to compute the overhead estimate. We also insert debugging instructions to measure the number of decoy I/O instructions that are executed during the time interval at hand. The instructions in a loop are counted anew on each iteration.

Given the MIPS metric from Dhrystone, we calculate an estimate of the total amount of instructions that were executed during that time interval. We then calculate the fraction of those instructions that belongs to decoy I/O, which now becomes the fraction of 1 millisecond that was, on average, spent by the CPU to execute decoy I/O instructions. For example, if Dhrystone indicates a MIPS of 4800, and the time interval under consideration is 0.4 s, an estimate of the total number of instructions executed on the CPU is $192 * 10^7$. If the count of decoy I/O instructions is 384, and taking into account that 1 millisecond is equal to 10^6 nanoseconds, then the overhead estimate pertaining to the time interval at hand is $\frac{384}{192 * 10^7} * 10^6 = 0.2$ nanoseconds.

When the ICS machines are not under attack, the execution overhead of the decoy code fluctuates between 100 and 300 picoseconds at most. Most of this overhead is due to consistency measures. When under attack, the execution overhead increases up to under 1.5 nanoseconds due to the large volume of I/O requests and replies that the decoy code has to process. The various ICS malware analytics add to the execution overhead too. A sample of overhead measurements on a protective relay is given in Fig. 8. Execution overheads on other ICS machines are similar. These execution overheads are insignificant for protective relays and ICS machines in general.

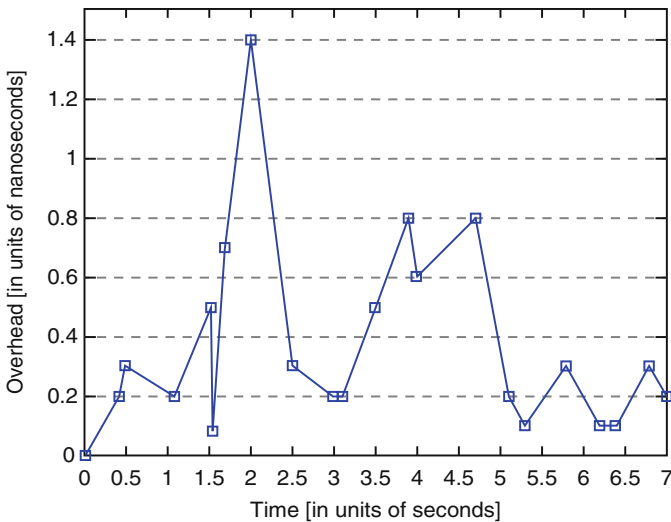


Fig. 8 A sample of execution overhead on a protective relay

7 Conclusions

Industrial mirage makes it highly complex for ICS malware to find their way to real segments of the electrical power grid without falling into a trap. ICS malware are detected on ICS machines in production on the very first encounter with them, and hence without any prior knowledge of their code and data. Decoy I/O of various forms characterize ICS malware in sufficient detail to enable ICS machines to isolate them and gradually recover from them entirely. Effective cyber operations can be enabled against ICS malware to trace and discover the human factors behind the attacks for the purpose of attribution. Industrial mirage is usable in practice, and has already demonstrated to be effective against the ICS malware samples that were used in recent campaigns. Further research will make industrial mirage absolutely safe to both legitimate applications and users on ICS machines, and the equipment and physics of the electrical power grid.

Acknowledgements This research is sponsored by the Air Force Office of Scientific Research and the U.S. Air Force Academy Center for Cyberspace Research under agreement number FA7000-16-2-0002. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force, Department of Defense, or the U.S. Government.

References

1. ICS-CERT: Cyber-attack against Ukrainian critical infrastructure. Available online at <https://ics-cert.us-cert.gov/alerts/IR-ALERT-H-16-056-01>
2. Lee RM, Assante J, Conway T (2016) Analysis of the cyber attack on the Ukrainian power grid. Defense use case white paper. Available online at https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf
3. Symantec (2014) Dragonfly: cyberespionage attacks against energy suppliers. Available online at https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/Dragonfly_Threat_Against_Western_Energy_Suppliers.pdf
4. Lezi S (2014) Scanderbeg, the hero of Europe. CreateSpace Independent Publishing Platform, Scotts Valley
5. Falliere N, Murchu LO, Chien E (2011) W32.Stuxnet Dossier. Symantec security response, version 1.4. Available online at http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf
6. Siemens: What properties, advantages and special features does the S7 protocol offer? Available online at <https://support.industry.siemens.com/cs/document/26483647/what-properties-advantages-and-special-features-does-the-s7-protocol-offer-?dti=0&lc=en-WW>
7. Homan J, McBride S, Caldwell R (2016) IronGate ICS malware – Nothing to see here... Masking malicious activity on SCADA systems. FireEye threat research Blog. Available online at https://www.fireeye.com/blog/threat-research/2016/06/irongate_ics_malware.html
8. Buza DI, Juhasz F, Miru G, Felegyhazi M, Holczer T (2014) CryPLH: Protecting smart energy systems from targeted attacks with a PLC honeypot. Smart grid security, vol 8448. Springer, Berlin, pp 181–192

9. Rist L, Vestergaard J, Haslinger D, De Pasquale A, Smith J, CONPOT ICS/SCADA honeypot. Available online at <http://conpot.org>
10. Vollmer T, Manic M (2014) Cyber-physical system security with deceptive virtual hosts for industrial control networks. *IEEE Trans Ind Inf* 10(2):1337–1347
11. International Electrotechnical Commission (2004) IEC 61850 – Communication Networks and Systems in Substations, parts 1 through 9
12. Rrushi J (2011) An exploration of defensive deception in industrial communication networks. *Int J Crit Infrastruct Prot* 4(1):66–75
13. Rrushi J (2016) NIC displays to thwart malware attacks mounted from within the OS. *J Comput Secur* 61(C):59–71
14. Simms S, Maxwell M, Johnson S, Rrushi J (2017) Keylogger detection using a decoy keyboard. In: *Proceedings of the 31st Annual IFIP WG 11.3 Conference on Data and Applications Security and Privacy*, Philadelphia. Springer, Cham
15. Rrushi J, DNIC architectural developments for 0-Knowledge detection of OPC malware. Currently in the second round of review at *IEEE Trans Dependable Secure Comput*
16. Lange J, Iwanitz F, Burke T (2010) *OPC – from data access to unified architecture*, 4th edn. VDE Verlag GmbH, Berlin
17. International Organization for Standardization, Technical Committee 184: manufacturing message specification. Available online at <https://www.iso.org>
18. RTDS Technologies: real time digital power simulator. Available online at <https://www.rtds.com>
19. Strogatz SH (2014) *Nonlinear dynamics and chaos – with applications to physics, biology, chemistry, and engineering*, 2nd edn. Westview Press, Boulder
20. Ott E (2002) *Chaos in dynamical systems*, 2nd edn. Cambridge University Press, Cambridge
21. Ott E, Grebogi C, Yorke JA (1990) Controlling chaos. *Phys Rev Lett* 64(1196):1196–1199
22. Romeiras F, Grebogi C, Ott E, Dayawansa WP (1992) Controlling chaotic dynamical systems. *Phys D* 58(165):165–192
23. Searcy W, Nowicki S (2005) *The evolution of animal communication – reliability and deception in signaling systems*. Princeton University Press, Princeton
24. Goldberg DE (1989) *Genetic algorithms in search, optimization and machine learning*. Kluwer Academic Publishers, Boston
25. Brogan WL (1990) *Modern control theory*, 3rd edn. Prentice-Hall, Upper Saddle River
26. Simon D (2006) *Optimal state estimation – Kalman H infinity, and nonlinear approaches*, 1st edn. Wiley-Interscience, Hoboken
27. Fridrich J (2009) *Steganography in digital media – principles, algorithms, and applications*, 1st edn. Cambridge University Press, Cambridge
28. The Apache Software Foundation: Apache Hadoop. Available online at <http://hadoop.apache.org>
29. The Apache Software Foundation: MapReduce. Available online at https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html
30. Lie D, Thekkath CA, Mitchell M, Lincoln P, Boneh D, Mitchell JC, Horowitz M (2000) Architectural support for copy and tamper resistant software. In: *Architectural Support for Programming Languages and Operating Systems (ASPLOS IX)*. ACM, New York, pp 168–177
31. Chen B, Morris R (2003) Certifying program execution with secure processors. In: *Proceedings of the Usenix Workshop on Hot Topics in Operating Systems*. Lihue, Hawaii
32. DNP Technical committee: distributed network protocol. Available online at <https://www.dnp.org>

Flood Resilience of a Water Distribution System



Fabio Tarani, Chiara Arrighi, Laura Carnevali, Fabio Castelli, and Enrico Vicario

Abstract Extreme weather events such as heavy rains and floods are becoming more frequent and severe due to global warming, therefore leading to an increasing interest in methods to evaluate environmental consequences and mitigation strategies. Water supply systems (WSS) represent a class of safety-critical infrastructure prone to damage, with direct impact on public health. They can be cast in the class of cyber-physical systems, since their operation is governed by their physical behaviour—related to topology, fluid-dynamics and technology—which in turn is steered by operation policies and user behaviour—pump and valve management, demand–response mechanisms, etc. In this context, we propose an approach to estimate resilience in the indirect damage caused by a flood on a Water Supply System (WSS). To this end, we combine analysis of an inundation model, which computes the floodwater depth over time on the studied territory, and evaluation of a hydraulic network model by a Pressure-Driven Demand (PDD) approach, which also allows for demand–response mechanisms. Flood damage is assessed in terms of both lack of service experienced by inhabitants and length of pipeworks contaminated by floodwater. The approach is experimented on the WSS of Florence, Italy, which serves about 380,000 users and lies in a flood-prone territory. A sensitivity analysis is with respect to demand–response efficiency, speed, and start time.

Keywords Water distribution networks · Flood · Resilience · Hybrid systems

F. Tarani (✉) · L. Carnevali · E. Vicario
Department of Information Engineering (DINFO), Università di Firenze, Firenze, Italy
e-mail: fabio.tarani@unifi.it; laura.carnevali@unifi.it; enrico.vicario@unifi.it

C. Arrighi · F. Castelli
Department of Civil and Environmental Engineering (DICEA), Università di Firenze,
Firenze, Italy
e-mail: chiara.arrighi@dicea.unifi.it; fabio.castelli@unifi.it

1 Introduction

Extreme weather events and natural disasters are raising increasing concerns worldwide, also due to climate change perspectives [10, 15, 25]. In particular, interest is spreading on the consequences on population [2], environment [6], urban areas and, infrastructures [7, 13, 19, 22]. In this context, the estimation of flood impact opens new research perspectives, as shown by the sustainability criterion adopted by the flood risk mitigation strategies of the EU Parliament [9], which promotes quantitative flood risk assessment [18] and flood damage maps [8, 19].

Water Supply Systems (WSS, see Table 1 for a list of acronyms frequently used in the chapter) are safety-critical network infrastructures as a main factor in environmental sustainability, public health, and resilience [14, 24]. Their behaviour is intrinsically cyber-physical due to the interaction of management of operation, which lies on an abstract, information-based layer, with a physical system ruled by partial differential equations. In this context, the increasing availability of smart metering and control devices will increase the influence of cybernetic control aspects on network operation, for example by directly managing demand or allowing a finer control on network operation. Moreover, WSS can be cast in the class of Stochastic Hybrid Systems (SHS) [1, 4, 16] due to their continuous behaviour undergoing sudden changes at stochastic times due to a number of discrete variables, related for example to pump operation schedules or pressure-controlled sectioning valves.

Due to their great spacial extent and the functional interdependencies among their components, WSS and other network infrastructures propagate the effects of contingencies affecting some part of the network farther from the location of the event. Hence, the detrimental consequences of a contingency are usually classified into direct and indirect damage. In the case of inundations, the former relates to physical contact with floodwater, accounting for instance for damaged equipment or contaminated pipes, and is often estimated by means of damage curves, whereas the latter refers to the impact outside the flooded area, e.g. to the number of inhabitants experiencing lack of service even if located far from the flooded areas.

Management of flood risk entails a combined approach comprising mitigation, preparedness, response, and recovery [24], a fundamental role being played by hazard identification and vulnerability analysis. Impact assessment is a valid support for decision makers: in particular, being able to identify impacts and to

Table 1 Acronyms used in this chapter

Acronym	Definition
DTM	Digital Terrain Model
DWTP	Domestic Water Treatment Plant
LIDAR	Light Detection and Ranging
PDD	Pressure-Driven Demand
RI	Recurrence Interval
WDN	Water Distribution Network
WSS	Water Supply System

estimate the effects of various mitigation strategies is fundamental both in a tactical and strategic outlook: from a tactical perspective, an optimal exploitation of available resources can be achieved during the emergency phase, e.g. by controlling service levels and providing appropriate backup systems, whereas on the long-term, strategic run, a better allocation of budget can be attained in terms of resilience maximisation.

In this chapter, the impact of a flood on a WSS in terms of indirect damage is evaluated. The implemented method exploits an automated procedure integrating, by means of a GIS infrastructure, an inundation model, and a hydraulic network model with Pressure-Driven Demand (PDD) [5, 21]. Two measure for the assessment of flood impact are introduced to quantify functional deficit—in terms of lack of service—and structural damage—in terms of pipe contamination. The method is tested on a real case study, namely the WSS of Florence, Italy, which serves approximately 380,000 inhabitants and lies in a flood-prone area.

The remainder of the chapter is organized as follows: in Sect. 2, the application context is described in detail; in Sect. 3, the model, its constituent sub-models, and the defined measures are presented; in Sect. 4, the calculated measures are shown and sensitivity analyses are performed on the key input parameters. Finally, conclusions are drawn in Sect. 5.

2 Application Context

The fresh water supply chain comprises several phases: abstraction from source, transport, treatment, and distribution. The last two operations are performed by means of WSS, which typically include one or more treatment plants, where water conveyed from the source is processed and made biologically and chemically safe, and a set of pipes connected in a network, which delivers it to users.

The network can be abstracted as a graph, where nodes represent junctions and users (demand nodes), whereas edges correspond to pipes. Each node is characterised by an elevation and a time-varying nodal demand, which depends on the type of associated users (domestic, office, industry, etc.) and is null for junction nodes. Each pipe has a set of attributes which may include inner diameter, length, material, age, and minor loss coefficient.

Special nodes are used to represent tanks and reservoirs, whereas pumps and valves (sectioning, pressure-control, etc.) are more easily represented as edges. Tanks are used to provide storage capacity and enhance stability of operation by decoupling demand (largely time-varying) and production rate (usually constant). They are often filled by a dedicated lifting station which is turned on either on a predefined schedule or based on a pressure measurement in the network. Their effect on network operation is twofold. On the one hand, they lead to a hybrid behaviour because their finite capacity leads to discrete changes in their working condition even if the dynamics are generally continuous. On the other hand, they introduce memory, thus ruling out strategies based on steady-state analysis. For what concerns

hybrid behaviour, a tank can be in one of three states: in operation, full, or drained. When it is in operation, water can freely enter or leave the tank according to pressure and mass balances; when full, water can only exit the tank, so that, if water head in the network is higher than in the tank (i.e., the network is pushing water inside the tank), the latter is considered as a regular junction node (no demand) and allows no water inlet. A similar reasoning applies when the tank is drained and water can enter the tank if required but not leave it.

Special edges such as pumps—used to maintain required heads, to fill elevated tanks and to displace water throughout the network—and valves—employed for flow control and pipe sectioning—also contribute to the hybrid behaviour of the system and strengthen the cybernetic aspects. Even if variable-speed motors are increasingly available, most pumps are controlled in a binary fashion, being turned off and on according either to a schedule or to a level/pressure measurement. Valves, on their part, can either be intrinsically discrete (e.g., backflow valves) or they can be controlled by the network operator, either locally or remotely.

The large number of special components in real networks—even a medium-sized network may contain several tanks, pumps and valves—and the intrinsic non-linearity of the partial differential equations governing pressure loss in pipes lead to a great complexity of the problem, ruling out linear solution methods and calling for specifically designed analysis techniques [17]. Most used techniques involve discretisation on the time-domain and iterative solution of the non-linear system representing the system state at each timestep, for example by Newton-Raphson methods. A set of checks is also necessary to promote stability of the solution and ensure convergence.

The presented methodology is applied to the urban area of Florence, Italy, with an extension of 102 km², which is crossed by the river Arno whose bed, extending for 8 km, conveys an average flowrate of 60 m³s⁻¹. Notwithstanding the broadness of its catchment area (8200 km²), the low permeability of its territory leads to a highly torrential flow pattern, with recorded flow rates as low as 1 m³s⁻¹ and as high as 4100 m³s⁻¹. Due to such variability, the area is extremely flood-prone, fact witnessed by historical records which mention over 150 such events in the last millennium.

For what concerns the WSS, the city is served by a treatment plant which lies next to the upstream section of the river and abstracts in regular operation an average flowrate of 1.4 m³s⁻¹, which is treated and pumped into the distribution network by three 710-kW pumps. The network features over 900 km of pipeworks and 17 tanks, mostly located on the hills around the city, whose total storage capacity of 42,000 m³ is used to enhance stability of supply.

3 Methodology

The methodology for the evaluation of indirect flood damage interleaves the analysis of two models: an *inundation model*, which is used to predict the time-varying floodwater depth on the studied territory in case of flood, and a *WSS model*, which

accounts for the behaviour of the WSS during the event. In the following paragraphs, both models are presented in detail together with the overall data flow.

3.1 General Structure

The methodology is based on the following considerations regarding the behaviour of the WSS. Following common practices enabled by the increasing availability of smart metering and control devices and technologies, we consider demand–response measures activated so as to limit water consumption in case of a flood forecast. Specifically, these measures may include issuing a warning to the population asking to limit water consumption or actual throttling or closing on terminal pipes by means of smart devices. Starting from a certain time t_{DR} , consumption is reduced exponentially with respect to regular operation till an asymptotic reduction factor is achieved. Without loss of generality, the following equation is used to calculate requested flowrate of each user at time t :

$$D_{req,i}(t) = \begin{cases} D_{nom,i}(t) & \text{if } t \leq t_{DR} \\ D_{nom,i}(t) [(1 - DRE) + DRE e^{-(t-t_{DR})/\tau_{DR}}] & \text{if } t > t_{DR} \end{cases} \quad (1)$$

where $D_{req,i}(t)$ is the demand required by node i at time t , $D_{nom,i}(t)$ is the corresponding nominal demand, DRE is the total Demand Response Effectiveness, and τ_{DR} is the time constant of the demand response process. The demand–response behaviour is thus defined by three parameters: t_{DR} , τ_{DR} and DRE.

After the event begins, floodwater affects WSS operation in a twofold way: on the one hand, it leads to failure of exposed active components, e.g. by impairing power supply or by damaging essential subsystems; on the other hand, it leads to pipe contamination.

- As regards failure of components, in a first approximation and lacking more detailed data on mechanical and electrical layouts, it is assumed that all electrically-powered devices are failure-prone, and that failure is triggered by the floodwater level with respect to the ground. A constant threshold of 0.5 m is defined, i.e. the device fails when water level exceeds this value, at time t_{fail} . This approach can easily be refined if more data about installation geometry is available.
- Pipe contamination occurs where floodwater head exceeds freshwater head inside the pipe. When this happens, contaminated water from the ground containing potentially harmful substances or bacteria may enter the network through junction clearances or small cracks, which are always present to some extent in real networks, therefore requiring thorough cleaning before the affected pipe is used again to deliver drinking water.

Figure 1 shows the general structure of the methodology. First, either a hydro-meteorological model, taking into account weather and land morphology, or

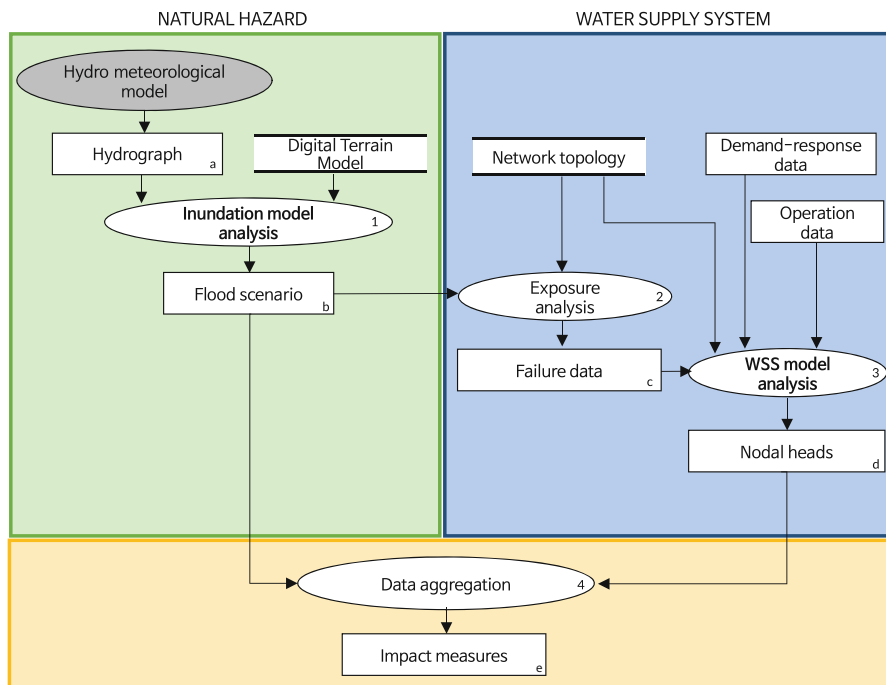


Fig. 1 Data flow diagram of the methodology. Greyed-out background indicates that the activity is not part of the methodology

empirical data are used to generate an hydrograph for the river (a), i.e. to estimate the flowrate conveyed by the river as a function of time following a period of exceptional precipitations on its catchment area. Hydrographs typically feature an hill-shaped pattern, with flow rate rising rapidly from a base level to a peak value and then slowly reverting to the initial value. Afterwards, the Digital Terrain Model (DTM) and the hydrograph are used in the inundation model (1) to compute transient floodwater levels (b). An exposure analysis (2) is then performed to obtain a list of failing devices and their failure times (c). Computed failure times, network topology, operation data (demands, pumps schedules and rules, etc.) and assumptions on the demand–response patterns are used in the WSS model (3) to evaluate transient nodal heads (d). Hydraulic head (henceforth abbreviated as “head”) is a measure of the pressure of a fluid expressed in length units, and is commonly employed in the field of WSS. Nodal heads, together with floodwater levels, are used by the data aggregation procedure (4) to determine measures of interest (e).

It should be pointed out that for a correct interaction of the models, items and events must be consistently located in space and time, respectively. To this end, a GIS framework is used to reference both the DTM and the network topology, whereas the start time of the hydrograph, coinciding with the arrival of the

flood-wave, is taken as reference time for all events. In particular, demand–response start time t_{DR} can also be negative (procedures activated preventively), whereas failure time t_{fail} is always positive.

3.2 *Analysis of the Inundation Model*

The inundation model shall take a hydrograph and a DTM as inputs and returns a flood scenario as the output. Among the many available tools, the Telemac-Mascaret framework has been chosen to simulate flood dynamics during the event. Telemac-Mascaret is an integrated suite of solvers widely used for free-surface flow studies, and is de facto one of the major standards in its field, developed and managed by a consortium of industries and research organisations, and open source [11]. The simulation modules use algorithms based on the finite-element method [12]. Space is discretised in the form of an unstructured grid of triangular elements, which means that it can be refined particularly in areas of special interest.

The module used for the purpose of this research solves the Navier-Stokes Partial Differential Equations (PDE) for liquid flow, taking into account both sub- and super-critical flows and ground friction. Geographic data is supplied as a 2-D triangular mesh representing the topography of the studied area with an elevation value assigned to each node. On the boundary nodes of the mesh, specific flow conditions must be given. In particular, three types of boundary conditions are chosen for the riverine flow analysis: the upstream boundary of the riverbed is assigned time-varying free surface elevation and flow rate profiles, the downstream border is assigned a fixed elevation for stability issues, whereas the remainder of the boundary is assumed closed (no throughflow). All of them are simplified conditions which must be carefully supervised as not to affect the reliability of results. In particular, the most stringent condition refers to the outflow boundary: as long as the flow regime along such border is supercritical no issue emerges, since downstream conditions cannot affect the studied domain. If the flow is subcritical, however, some information on the flow capacity of the downstream area should be provided for increased precision. In the present study, a fixed-elevation, free-flowrate regime has been used to improve computation stability. A sensible elevation on the outflow boundary has been chosen, keeping in mind that estimated water depth near such border is subject to a higher error than the remainder of the domain.

In summary, the implemented inundation model takes the following inputs: (1) land topography as a triangular unstructured 2-D mesh with elevation values, (2) time-varying inlet flow rate at upstream boundary, and (3) elevation at downstream boundary. The time-varying water depth at each node is returned as an output. Even if a time-continuous, space-continuous system is analysed, the model returns water depths for a finite number of nodes at a finite number of timesteps. If water depth is needed for points not belonging to the mesh at other times (e.g. to evaluate floodwater depth on a WSS component with given coordinates), interpolation in space and time must be used.

3.3 Analysis of the WSS Model

The WSS model takes network topology, demand–response data (nodal demands) and operation data (schedules, failed components) as inputs and returns nodal heads throughout the network as output. The behaviour of the WSS is simulated by means of EPANET [20], a software that models water distribution piping systems, developed by the United States Environment Protection Agency (EPA) and freely distributed. The tool solves flows in pipe networks using mass conservation and the Hazen-Williams equations for head loss, performing quasi-static time-varying simulations also implementing tank dynamics. The hybrid behaviour introduced by discrete-dynamics components such as time- or level-switched pumps is also effectively taken into account. In particular, EPANET calculates time-varying pressures at the nodes given a set of initial tank levels, pump switching criteria, node base demands and demand patterns.

The main hindrance to using the standard EPANET implementation for the purposes of current research is its strict demand-driven approach, which stems from the primary goal of simulating correctly-operated networks. In such networks, pressure at each node is sufficient so as to allow withdrawal of required demand from each node, so that demands can be assumed as defined input data. However, when simulating strongly off-design networks, nodes featuring a reduced pressure are common, so that a pressure-driven approach is needed [5, 23]. PDD models differ from conventional ones in that demands at nodes are not attributed a priori, but their value depends on the current pressure at the considered node.

As done in [3] consistently with practice, in the presented implementation each node can be in one of three states:

fully served if the node is able to withdraw its nominal demand, i.e. when $H_i(t) \geq H_{\text{service}}$, where $H_i(t)$ is nodal head at time t and H_{service} is the minimum head to be considered fully served; the served demand of node i , D_i , is therefore equal to its required demand: $D_i = D_{\text{req},i}$;

partially served if the node experiences lack of pressure ($H_{\text{service}} > H_i(t) > 0$) and withdraws a reduced demand; this reduced demand is expressed as

$$D_i = D_{\text{req},i} \left(\frac{H_i}{H_{\text{service}}} \right)^\alpha \quad (2)$$

where α is a constant exponent deriving from the physics (in particular from the relationship between kinetic energy and velocity) and set to 0.5;

non served if pressure is null ($H_i(t) = 0$) and the node is unable to withdraw any water, yielding null served demand ($D_i = 0$).

In order to allow for such behaviour, the standard EPANET must be modified. In fact, EPANET only allows two types of nodes: *nodes* are assigned a time-varying, pressure-independent demand, and can be effectively used to model fully served users, whereas *emitters*, conceived to model fixed cross-section water outlets such as fire hoses and orifices, adequately model the aforementioned behaviour


```

1: procedure PDDEPANETRUN(nodeStates)
2:    $t \leftarrow 0$ 
3:   while  $t < \text{Duration}$  do
4:     INITIALIZENODESTATES(nodeStates)
5:     RUNEPANET(nodeStates)
6:     while ARENODALHEADSNOTCONSISTENT(nodeStates) do
7:       UPDATENODESTATES(nodeStates)
8:       RUNEPANET(nodeStates)
9:     end while
10:    INCREASETIME(t)
11:  end while
12: end procedure

```

Fig. 2 Pseudocode for the MATLAB implementation

of partially served users. Emitters are defined by a fixed exponent α , equal for all instances, and a flow coefficient K_i which represents the volume flow rate for unitary pressure loss across the orifice. In detail, flow rate through emitters is computed as:

$$D_i = K_i H_i(t)^\alpha \quad (3)$$

where K_i is the emitter flow coefficient, $H_i(t)$ is the head at time t and α is the emitter exponent, equal to 0.5 due to the physics. Unfortunately, emitters do not cope well with calculated negative pressures, attributing a negative (entering) flow rate where such negative pressures occur.

A MATLAB code has been implemented so as to exploit the aforementioned characteristics to run transient analysis while correctly using a PDD approach. The code—as shown in pseudocode Fig. 2—works as follows: three node states are defined: “2” for served nodes, “1” for partially served ones, and “0” for non-served ones; type 2 and type 0 nodes are modelled as EPANET nodes with nominal demand equal to the assigned nominal demand $D_{\text{req},i}$ and 0 respectively, whereas type 1 nodes are modelled as emitters whose flow coefficients are calculated to ensure that $D_i = D_{\text{req},i}$ if $H_i = H_{\text{service}}$. For each timestep, a first trial simulation is run with all nodes in state 2 in order to get the expected pressures. Afterwards, each node is checked to assess whether its pressure is in the pressure range corresponding to the current flow regimen and, if this is not the case, its state is accordingly raised or lowered by one unit (namely, it is not possible to jump from state 2 to state 0 and vice versa). After node states have been changed, simulation is repeated till no more node state change is necessary. Calculated flow rates and pressures are considered to represent network operation during the following timestep. In particular, flow rates are used to calculate the time to the next event (tank being filled or emptied), and the first event affecting network topology is considered (e.g. demand change, or pump setting toggle due to time pattern, tank getting empty or full). Tank levels are then updated and simulation proceeds to the next timestep.

The described procedure allows to calculate head and supplied demand at each node for each timestep, therefore fully estimating the network state in each moment.

3.4 Definition of Measures

Two measures have been defined in order to evaluate functional and structural impact of the flood on the network.

First, impact of the flood on operation is assessed by estimating the fraction of the required demand not met by the WSS.

$$\text{DNM}(t) = 1 - \frac{D_{\text{met}}(t)}{D_{\text{req, TOT}}(t)} = 1 - \frac{\sum_i D_i(t)}{\sum_i D_{\text{req},i}(t)} \quad (4)$$

where $D_{\text{met}}(t)$ is the total met demand at time t and $D_{\text{req, TOT}}(t)$ is the total required demand at time t , as defined in (1).

As a second measure, network damage due to pipe contamination is evaluated by calculating the total length of pipework to be decontaminated. A pipe is considered to be contaminated at time t if at any preceding moment in time the head inside the pipe is lower than the floodwater head outside, or below zero:

$$L(t) = \sum_{i \in I} \sum_{j \in J_i} L_j \quad \text{with } I = \{i \mid \exists \tau < t \text{ s.t. } H_i(\tau) < H_{\text{flood},i}(\tau)\} \quad (5)$$

where J_i is the set of pipes with either end connected to node i and L_j is the length of pipe j . $H_{\text{flood},i}(t)$ is the floodwater head on the node, and is obtained by interpolating the data given in output by the inundation model so as to match the geographical location of the node.

4 Experimental Results

The proposed methodology is implemented in a toolchain, using input data referring to the urban area of Florence, Italy, for a flooding event with Recurrence Interval (RI) 200 years and duration 18 h. The two defined measures have been calculated during the day of the event and the following, and the sensitivity to the three parameters defining demand–response effects (DRE, t_{DR} , τ_{DR}) has been evaluated. In the following, the employed data is described and the obtained experimental results are discussed.

4.1 Flood Dynamics

The mesh needed by the inundation model has been obtained by interpolation on data from a laser altitude (Laser Imaging Detection and Ranging—LIDAR) acquisition campaign promoted by the City Administration (*Comune di Firenze*).

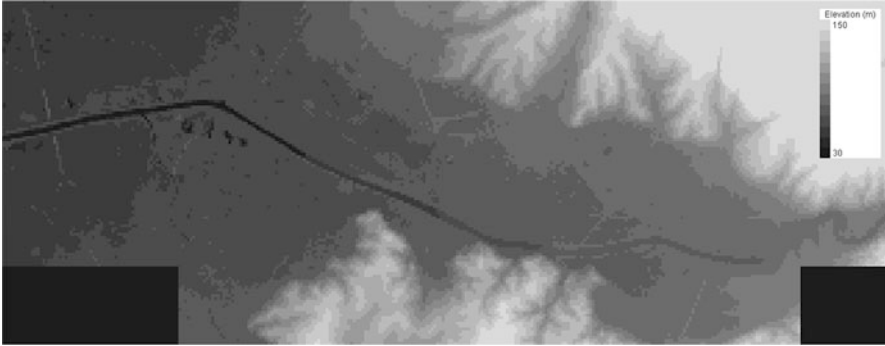


Fig. 3 Elevation data of the studied spacial domain, a subset of the urban area of Florence, Italy. Darker areas in the lower corners lie outside administrative borders and are not available

Used elevation data are collected on a regular square grid with 10 m spatial resolution. The generated triangular mesh features variable density: an average edge length of 10 m is used in an area extending 10 m on both sides of the riverbed, in order to better estimate the conveyable flowrate, whereas a coarser length of 40 m is used elsewhere, so as to improve computation times. Overall, the mesh includes 74,860 nodes, with elevation values ranging from 27 to 410 m (a contour plot of the elevation data for the analysed domain is shown in Fig. 3).

For what concerns the inflow, data have been provided by the river Catchment Authority (*Autorità di bacino del fiume Arno*). Data is made available for a set of scenarios, with RI ranging from 30 to 500 years and event duration from 3 to 36 h.

Simulation has been performed for the data based on the 200-year recurrence interval and 18-h event duration (the corresponding hydrograph is shown in Fig. 4). Simulated timespan is twice the event duration (36 h), with a calculation timestep of 2 s. Analysis has taken 180 mins on a quad-core i7 Intel CPU equipped with 8 GB RAM. The map in Fig. 5 shows in overlay the inundated areas at different times during the event.

The event leads to a maximum inundated area of 20.3 km², with a maximum water depth of 6.1 m. According to the previously mentioned failure criterion, pumping station failure is attained when water depth at the corresponding node reaches 0.5 m, i.e. 8.2 h after the beginning of the event.

4.2 Water Supply System Dynamics and Sensitivity Analysis

Network topology and component data for the EPANET model have been provided by the company managing the WSS (*Publiacqua SpA*). The slightly skeletonised model provided includes 4880 nodes, 6336 pipes, 17 tanks and 16 pumps, with a total pipe length of over 600 km.

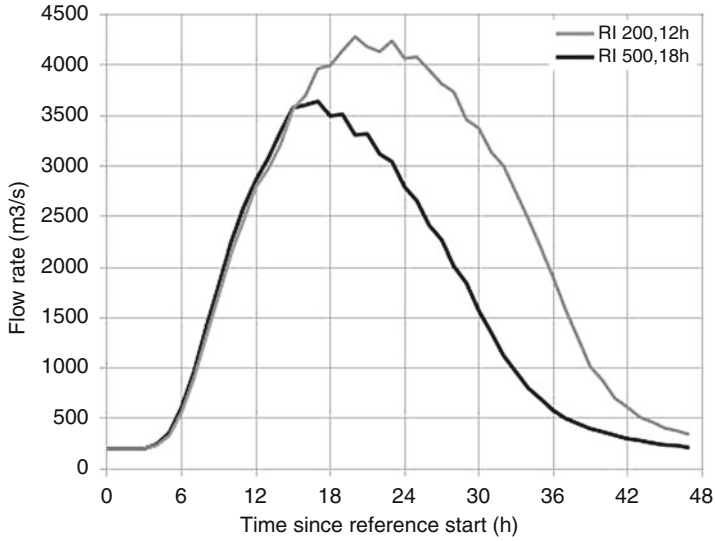


Fig. 4 River hydrograph (flow rate vs. time) for a flood scenario with 200-year recurrence interval and 18-h duration

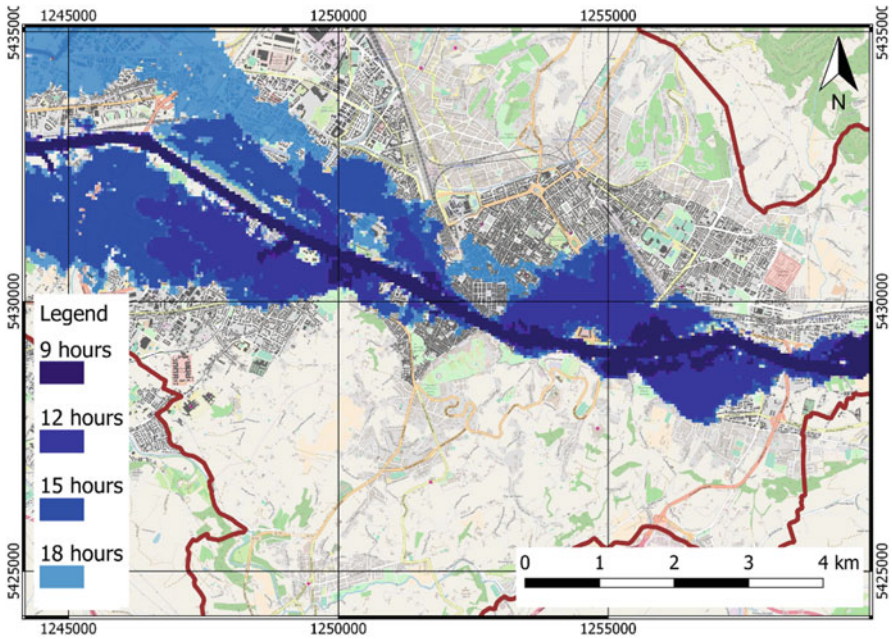


Fig. 5 Flooded areas at different times after the arrival of the flood wave

The model has been exploited to evaluate the influence of some input parameters on the evolution of the defined measures. In particular, temporal parameters defining the demand–response mechanism were varied to appreciate their influence on the evolution of Demand Not Met (DNM) and contaminated length L . The flood wave is assumed to start at midnight, and WSS failure occurs accordingly at 8 am. Measures are calculated, starting from midnight, during 48 h, thus covering two whole days. Recovery transient is not evaluated in the work, but it may be assumed that, starting from the time at which full treatment capacity is restored, (1) L stops rising, while (2) DNM quickly drops to zero, even if delivered water may not be biologically safe till pipe decontamination has been carried out. In this perspective, estimated L helps the network operator to understand how critical the responsiveness to contingency is. Following paragraphs report the main results of the analysis.

4.2.1 Sensitivity to Demand–Response Efficacy

The efficacy of the demand–response measures DRE, as defined in (1), has been varied in the set $\{0.1, 0.3, 0.5\}$ and the results are shown in Figs. 6 and 7.

Figure 6 shows unmet demand DNM, as defined in (4), as a function of time (time is counted starting from midnight, two whole days are represented, and failure occurs at $t = 8$ h). DRE has a direct and strong influence on the service levels. Demand patterns are still recognisable in most cases, although their imprint decreases as DRE increases. Raising the DRE from 0.1 to 0.3 halves the unmet demand always but during the demand peaks, whereas in the 0.5 case the lack of service is almost unnoticeable. During the first day, the trend is descending for all curves: in this phase, the influence of the diminishing demand—due to the

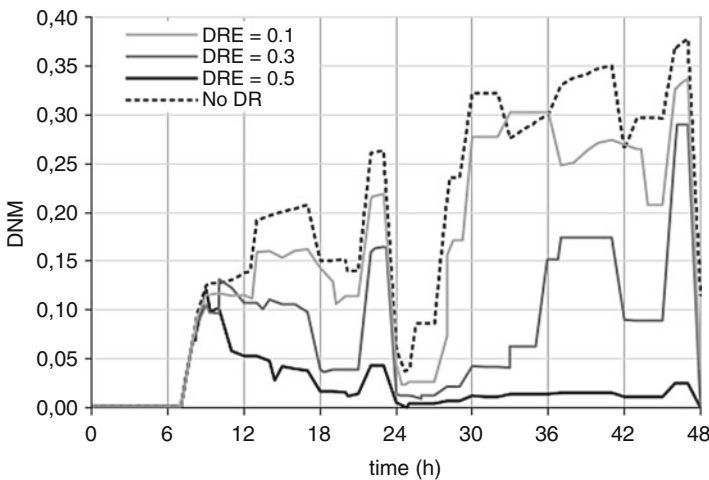


Fig. 6 Sensitivity of unmet demand on demand–response efficiency

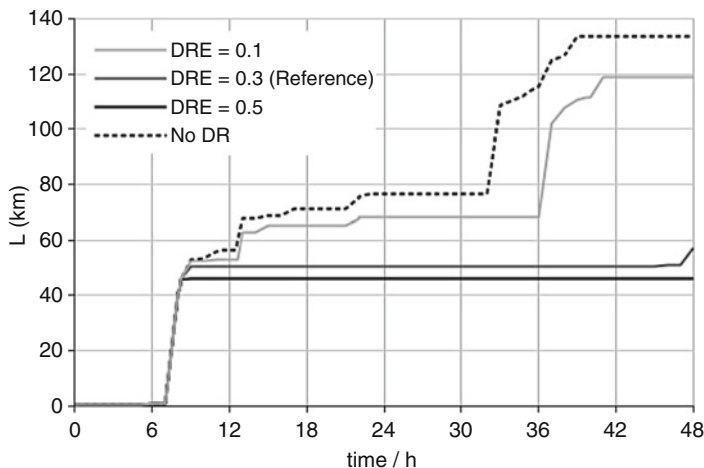


Fig. 7 Sensitivity of contaminated pipe length on demand–response efficiency

demand–response policies—prevails over the gradual draining of the tanks. The behaviour diverges during the second day: values keep rising in all cases except when $DRE = 0.5$. This happens because of the long-term flow balance: if the residual demand is lower than a threshold, the system is still able to supply most user notwithstanding its reduced production capacity, thus the unmet demand is very low. On the contrary, if production is lower than reduced demand, tanks will keep emptying, so that in the long-term, when all previously stored water has been supplied, most users will experience lack of service.

Figure 7 shows the total length of contaminated pipeworks L , as defined in (5). The measure rises quickly after the failure, and tends to keep rising in discrete jumps. This is related to the hybrid behaviour of the system due to draining tanks, which cannot provide adequate head in some zone of the network. For higher values of DRE, jumps occur later in time so that the measure rises more slowly. Overall, contamination affects less than 20% of the network in the worst case (no demand–response policy).

4.2.2 Sensitivity to Demand–Response Time Constant

Figures 8 and 9 show the influence of the demand–response policy time constant, varied in the set $\{3, 6, 9\}$ h, on the defined measures. Again, the speed of the reaction positively affects operation, especially in the first hours after the failure. During day 1, for example, lack of service can be halved by doubling the policy speed ($\tau_{DR} = 3$ h). The effect is much less noticeable at the end of day 2, when demand reduction has attained its asymptotic value independently of the chosen time constant.

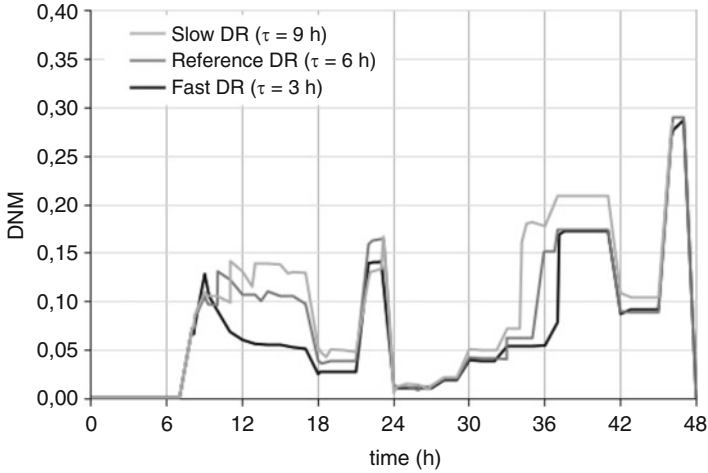


Fig. 8 Sensitivity of unmet demand on demand–response speed

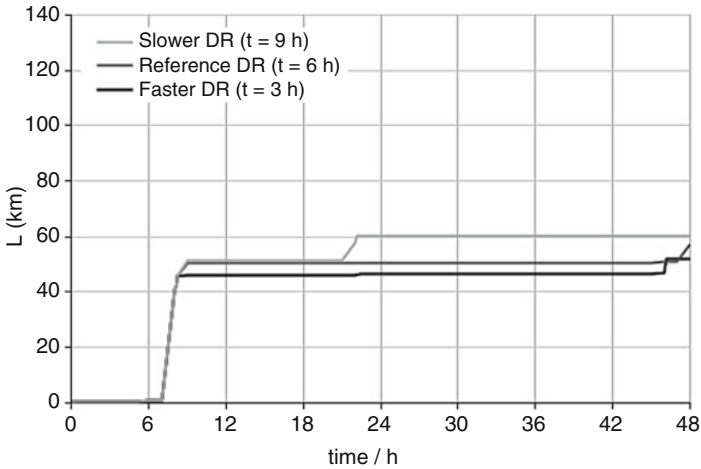


Fig. 9 Sensitivity of contaminated pipe length on demand–response speed

Figure 9 shows that the demand–response time constant slightly affects pipe contamination: a slower demand–response leads to less consumption and delays the time at which tanks drain.

4.2.3 Sensitivity to Demand–Response Start Time

Figures 10 and 11 show the influence of the demand–response policy start time, varied in the set $\{-4, 0, +4\}$ h with respect to failure time, on the defined measures.

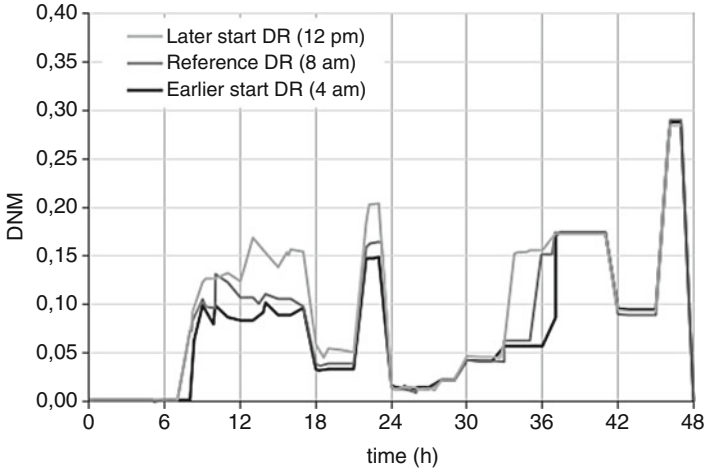


Fig. 10 Sensitivity of unmet demand on demand–response start time

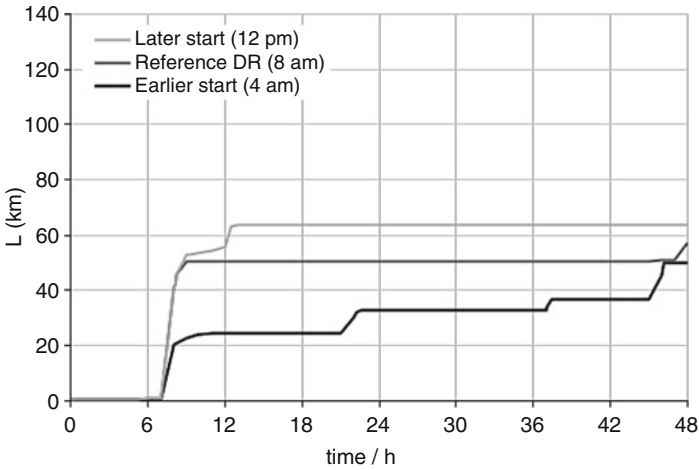


Fig. 11 Sensitivity of contaminated pipe length on demand–response start-time

The influence of the start time on both measures is stronger during the first hours after failure. Delaying start time by 4 h, from 8 o'clock (time of failure) to midday, increases unmet demand by 50% during the first 10 h after failure. The effect of start time is barely evident during day 2, due to the steady-state condition attained in demand reduction. For what concerns contaminated pipe length, anticipating demand response leads to a lower consumption during peak hours and thus to a slower effect on tank draining and pipe contamination.

5 Conclusions

Water supply systems are cyber-physical systems with critical impact on public safety and prone to failure in case of flood. In this chapter, we have presented a methodology to evaluate the indirect impact of a flood on a water supply system. The approach leverages the analysis of an inundation model to predict floodwater levels in the transient phase following the extreme event with a hydraulic network model. In particular, the computed floodwater levels enable an exposure analysis to forecast failure times of critical devices, which, in turn, are used in the analysis of the hydraulic model of the WSS to derive measures of the flood impact, both in terms of demand not met and in terms of pipe contamination.

Experiments performed on the WSS of Florence, Italy, prove feasibility and effectiveness of the approach on a real case. Notably, the results of a sensitivity analysis with respect different demand–response parameters can be used to compare different strategies for reduction of water consumption, supporting decisions of WSS operators.

Acknowledgements We acknowledge the network operator Publiacqua SpA for the WSS hydraulic model of the city of Florence, Italy.

References

1. Abate A, Katoen J, Lygeros J, Prandini M (2010) Approximate model checking of stochastic hybrid systems. *Eur J Control* 16(6):624–641
2. Ashley ST, Ashley WS (2008) Flood fatalities in the United States. *J Appl Meteorol Climatol* 47(3):805–818
3. Arrighi C, Tarani F, Vicario E, Castelli F (2017) Flood impacts on a water distribution network. *Natural Hazards Earth Syst Sci Discussions* 2017:1–22. [Online]. Available: <https://www.nat-hazards-earth-syst-sci-discuss.net/nhess-2017-205/>
4. Bujorianu ML, Lygeros J (2004) General stochastic hybrid systems: modelling and optimal control. In: *Proceedings IEEE Conference Decision Control*, Nassau, Bahamas, pp 1872–1877
5. Cheung P, Van Zyl J, Reis L (2005) Extension of epanet for pressure driven demand modeling in water distribution system. *Comput Control Water Ind* 1:311–316
6. Christodoulou SE (2011) Water resources conservancy and risk reduction under climatic instability. *Water Resour Manag* 25(4):1059–1062
7. Emanuelsson MAE, McIntyre N, Hunt CF, Mawle R, Kitson J, Voulvoulis N (2014) Flood risk assessment for infrastructure networks. *J Flood Risk Manage* 7(1):31–41
8. de Moel H, van Alphen J, Aerts JCJH (2009) Flood maps in Europe – methods, availability and use. *Nat Hazards Earth Syst Sci* 9(2):289–301
9. EU Parliament (2007) Directive 2007/60/EC, pp 27–34
10. IPCC (2013) Summary for policymakers. *Climate change 2013: the physical science basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*, p 33
11. Galland J, Goutal N, Hervouet J (1991) A new numerical model for solving shallow water equations. *Adv Water Resour* 14(3):138–148
12. Hervouet J (2007) *Hydrodynamics of free surface flows modelling with the finite element method*, vol 1. Wiley, Hoboken

13. Khan SJ, Deere D, Leusch FDL, Humpage A, Jenkins M, Cunliffe D (2015) Extreme weather events: should drinking water quality management systems adapt to changing risk profiles? *Water Res* 85:124–136. [Online]. Available: <http://doi.org/10.1016/j.watres.2015.08.018>
14. Luh J, Royster S, Sebastian D, Ojomo E, Bartram J (2017) Expert assessment of the resilience of drinking water and sanitation systems to climate-related hazards. *Sci. Total Environ* 592:334–344. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S004896971730596X>
15. Lung T, Lavalle C, Hiederer R, Dosio A, Bouwer LM (2013) A multi-hazard regional level impact assessment for Europe combining indicators of climatic and non-climatic change. *Glob Environ Chang* 23(2):522–536 [Online]. Available: <http://dx.doi.org/10.1016/j.gloenvcha.2012.11.009>
16. Lygeros J, Prandini M (2010) Stochastic hybrid systems: a powerful framework for complex, large scale applications. *Eur J Control* 16(6):583–594
17. Methods H, Walski TM, Chase DV, Savić DA, Grayman W, Beckwith S, Koelle E (2003) *Advanced water distribution modeling and management*. Haestad Press, Waterbury
18. Merz B, Kreibich H, Schwarze R, Thielen A (2010) Review article “assessment of economic flood damage”. *Nat Hazards Earth Syst Sci* 10(8):1697–1724
19. Meyer V, Becker N, Markantonis V, Schwarze R, Van Den Bergh JCJM, Bouwer LM, Bubeck P, Ciavola P, Genovese E, Green C, Hallegatte S, Kreibich H, Lequeux Q, Logar I, Papyrakis E, Pfurtscheller C, Poussin J, Przyłuski V, Thielen AH, Viavattene C (2013) Review article: assessing the costs of natural hazards-state of the art and knowledge gaps. *Nat Hazards Earth Syst Sci* 13(5):1351–1373
20. Rossman LA (1999) The EPANET programmer’s toolkit for analysis of water distribution systems. In: 29th Annual Water Resources Planning and Management Conference, Tempe, Arizona, United States, ASCE, Reston, VA, pp 1–10
21. Siew C, Tanyimboh TT (2012) Pressure-dependent EPANET extension. *Water Resour Manag* 26(6):1477–1498
22. Short MD, Peirson WL, Peters GM, Cox RJ (2012) Managing adaptation of urban water systems in a changing climate. *Water Resour Manag* 26(7):1953–1981
23. Walski T, Blakley D, Evans M, Whitman B (2017) Verifying pressure dependent demand modeling. *Proc Eng* 186:364–371. {XVIII} International Conference on Water Distribution Systems, {WDSA2016}. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1877705817313796>
24. WHO (2011) *Guidance on water supply and sanitation in extreme weather events*, p 132. [Online]. Available: <http://www.euro.who.int/en/health-topics/environment-and-health/water-and-sanitation/publications/2011/guidance-on-water-supply-and-sanitation-in-extreme-weather-events>
25. World Economic Forum (2017) *The Global Risks Report 2017 12th Edition Insight Report*

A Non-parametric Cumulative Sum Approach for Online Diagnostics of Cyber Attacks to Nuclear Power Plants



Wei Wang, Francesco Di Maio, and Enrico Zio

Abstract Both stochastic failures and cyber attacks can compromise the correct functionality of Cyber-Physical Systems (CPSs). Cyber attacks manifest themselves in the physical system and, can be misclassified as component failures, leading to wrong control actions and maintenance strategies. In this chapter, we illustrate the use of a nonparametric cumulative sum (NP-CUSUM) approach for online diagnostics of cyber attacks to CPSs. This allows for (i) promptly recognizing cyber attacks by distinguishing them from component failures, and (ii) guiding decisions for the CPSs recovery from anomalous conditions. We apply the approach to the Advanced Lead-cooled Fast Reactor European Demonstrator (ALFRED) and its digital Instrumentation and Control (I&C) system. For this, an object-oriented model previously developed is embedded within a Monte Carlo (MC) engine that allows injecting into the I&C system both components (stochastic) failures (such as sensor bias, drift, wider noise and freezing) and cyber attacks (such as Denial of Service (DoS) attacks mimicking component failures).

Keywords Cyber-physical system · Cyber attacks · Stochastic failures · Diagnostics · Nonparametric cumulative sum (NP-CUSUM) · Nuclear power plant · The authors would like to capitalize each word as Advanced Lead-cooled Fast Reactor European Demonstrator (ALFRED)

W. Wang · F. Di Maio (✉)
Energy Department, Politecnico di Milano, Milano, Italy
e-mail: francesco.dimaio@polimi.it

E. Zio
Energy Department, Politecnico di Milano, Milano, Italy

Chair on System Science and the Energy Challenge, Fondation Electricite' de France (EDF),
CentraleSupélec, Université Paris Saclay, Gif-sur-Yvette, France
e-mail: enrico.zio@polimi.it

Abbreviations

CPS	Cyber-Physical System
NP-CUSUM	Non-Parametric CUmulative SUM
ALFRED	Advanced Lead-cooled Fast Reactor European Demonstrator
I&C	Instrumentation and Control
MC	Monte Carlo
NPP	Nuclear Power Plant
PI	Proportional-Integral
DoS	Denial of Service
PID	Proportional-Integral-Derivative
FDI	False Data Injection
SG	Steam Generator
FA	Fuel Assembly
CR	Control Rod
SISO	Single Input Single Output
DAC	Digital-to-Analog Converter
LSB	Least Significant Bit

Nomenclature

P_{Th}	Thermal power
h_{CR}	Height of control rods
$T_{L,hot}$	Coolant core outlet temperature
$T_{L,cold}$	Coolant SG outlet temperature
Γ	Coolant mass flow rate
T_{feed}	Feedwater SG inlet temperature
T_{steam}	Steam SG outlet temperature
p_{SG}	SG pressure
G_{water}	Feedwater mass flow rate
G_{att}	Attemperator mass flow rate
k_v	Turbine admission valve coefficient
P_{Mech}	Mechanical power
$K_{p,j}$	Proportional gain value of j -th PI
$K_{i,j}$	Integral gain value of j -th PI
t	Time
t_R	Accident time
t_M	Mission time
Δt	Sensor measuring time interval
y	Variable (safety parameter)
y^{ref}	Reference value of controller set point value of y
$y^{real}(t)$	Real value of y

$y^{sensor}(t)$	Sensor measurement
$y^{feed}(t)$	Measurement received by the computing (feeding) subsystem
$y^{monitor}(t)$	Measurement received by the monitoring subsystem
$Y(t)$	Redundant channel measure, $Y = y^{feed}$ and $y^{monitor}$
$\delta_y(t)$	Sensor measuring error
$q_y(t)$	Converter quantization error
a	Accidental scenario
b	Bias factor
c	Drift factor
$S_Y(t)$	Score function-based statistic of the collected $Y(t)$, $S_Y(t) = S_y^{feed}(t)$ and $S_y^{monitor}(t)$
h_y	Positive threshold
τ_Y	Time to alarm, $\tau_Y = \tau_y^{feed}$ and $\tau_y^{monitor}$
$\Delta\tau_y$	Delay difference between τ_y^{feed} and $\tau_y^{monitor}$
Γ_y^{ref}	Reference delay difference
c_y	NP-CUSUM parameter
ε_y	NP-CUSUM parameter
ω_y	NP-CUSUM positive weight
g_Y	Score function
Δg_Y	Score function difference value
μ_Y	Pre-change mean value of Y
θ_Y	Post-change mean value of Y
$\widehat{\theta}_Y(t)$	On-line estimate of θ_Y
$\mu_{\Delta g_Y}$	Known pre-change mean value of Δg_Y
$\theta_{\Delta g_Y}$	Unknown post-change mean value of Δg_Y
α_y^h	False alarm rate
β_y^h	Missed alarm rate
$\gamma \left(\Gamma_{T_{L,cold}}^{ref} \right)$	Misclassification rate with respect to Γ_y^{ref}

1 Introduction

Cyber-Physical Systems (CPSs) feature a tight combination of (and coordination between) the system computational units and physical elements. To the benefit of safe operation, the integration of computational resources into physical processes is aimed at adding new capabilities to stand-alone physical systems, to enable functionalities of real-time monitoring, dynamic control and decision support during normal operation as well as in case of accidents. In CPSs, cyber and physical processes are dependent and interact with each other through feedback control loops (e.g., embedded cyber controllers monitor and control the system physical variables, whilst physical processes affect, at the same time, the monitoring system and the computation units by wired or wireless networks [2, 24, 27, 40]). The benefit of

such self-adaptive capabilities is the reason why CPSs are increasingly operated in energy, transportation, medical and health-care, and other applications [6, 23, 27]. In the context of nuclear energy, the introduction of digital Instrumentation and Control (I&C) systems allows Nuclear Power Plants (NPPs) to take advantage of the new technologies in the field, for safe operation [21].

In the context of CPSs, sensor measurements can be used to monitor the behavior of the systems under different operational conditions, including hazardous and malicious ones. Indeed, CPS functionality can be compromised by both hazards (safety related) and malicious threats (security related) [13, 26, 41, 65]. Hazards and cyber threats originate from different sources (stochastic degradations and accidental conditions, for the former, external malevolent activities that are usually less accessible and less predictable for the latter [4, 26]). Distinct properties and mechanisms between them suggest different assessment methodologies for their identification.

The difficulty lies in the fact that components hazards and malicious threats can lead to similar consequences on the system [25, 29, 45, 58]. For example, in a situation where system shutdown is demanded, both failure of the shutdown of the actuator and interception of the shutdown command by an attacker result in unavailability of the safety action. In such situation, diagnosing of the failure cause would allow taking the right decision to respond to the system shutdown unavailability with the right emergency procedure (e.g., manual operation of the actuation in the case of such cyber attack).

In this sense, diagnostic of cyber attacks and component failures is important for the system protection and resilience, allowing prompt recovery from the effects of disruptive events and, thus, increasing system resilience [14, 20, 35, 38, 69, 70].

In this work, we develop a nonparametric cumulative sum (NP-CUSUM) detection approach [44, 51–53] for diagnosing cyber attacks, distinguishing them from component failures. The proposed approach is illustrated considering the possible occurrence of stochastic components failures and cyber attacks in the digital I&C system of the Advanced Lead Fast Reactor European Demonstrator (ALFRED) [16]. An object-oriented simulator previously developed [42, 43], and comprising a multi-loop Proportional-Integral (PI) control scheme [49], is utilized for simulating the ALFRED dynamic response to failures and cyber attacks.

The main original contribution of this work lies in the prompt recognition and distinction of cyber attacks from component failures in CPSs by relying on the simultaneous treatment, within a consolidated NP-CUSUM approach, of the measurements taken from redundant channels, guiding decisions for the CPSs recovery from anomalous conditions.

The chapter is organized as follows. Section 2 sets the issue of security analysis in the framework of risk assessment and, highlights the contributions of cyber attack diagnostics to overall system resilience. Section 3 presents the main characteristics of the ALFRED reactor, with the data measuring and transmission, and control schemes in the channels of its digital I&C system at full power nominal conditions, and the MC engine for injection of component failures and cyber breaches. In Sect. 4, the proposed NP-CUSUM diagnostic algorithm is presented. The NP-

CUSUM diagnostic method is illustrated in Sect. 5 and evaluated with respect to its diagnostic performances, such as false alarm, missed alarm and misclassification rates in Sect. 6. Conclusions are drawn in Sect. 7.

2 Hazards and Threats for CPSs

CPSs demand that in the risk analysis both safety and security aspects are considered [12, 26, 41, 65]. With respect to safety, hazards relate to components failures that can result in accidental scenarios leading to unacceptable consequences on the system physical processes; as for security, malicious attacks can impair both the physical and cyber parts of the system, possibly leading to unacceptable consequences.

2.1 Hazards

Failures of both hardware and software can compromise the functionality of CPSs.

During operation, failures of embedded hardware components (e.g., sensors and actuators) can be induced by aging, degradation, and process and operational conditions, which modify the way components work and interact with each other, generating multiple failure modes [56]. For example, sensors can degrade and fail in different modes such as bias, drift and freezing [56]; actuators can fail stuck, accidentally driving the physical process to be isolated from the controlling units of the cyber domain [67, 71].

Components failures can lead to two types of misoperations: (1) failure on-demand, e.g., failing to trigger protections or execute proper control strategies (when demanded); (2) malfunction, e.g., spurious triggering of protections (e.g., unintentional shutdown) or incorrect execution of control actions. Failures on-demand and malfunctions of both hardware and software components have gained increasing attention in the risk community [1, 32].

Resilience of CPS to failures can be granted by self-adaptiveness of control decisions on actuators, resorting to intelligent control systems that properly manipulate sensors measurements [31]. For example, Proportional-Integral-Derivative (PID) controllers, typically used as feedback controller in CPS to retroact to actuators the actions to be undertaken for responding to changes of physical parameters, may suffer of software failures/errors (generated from inadequate specification, incomplete testing scope and algorithm/logic failures) that are latent and triggered only when context modifications are to be met [1, 22]. In these situations, control rules adaptability to variable physical conditions is a fundamental requirement to the robustness of CPS for resilience during CPS operation.

2.2 Threats

CPSs reliance on digitalization and remote control systems increases their exposure to cyber attacks to controllers, databases, networks and human-system interfaces, that can result in the loss of system functionality. Malicious activities can be manifested as Denial of Service (DoS) attacks [45, 47, 63, 66], False Data Injection (FDI) attacks (e.g., packet/data modification) [30, 34, 50], network scan & sniffing attacks [45, 55], integrity attacks (e.g., through malware contagion) [36, 37] and, illegal command executions [48]. They can be initiated in the cyber domain through local or remote accesses, mimicking the components failures but isolating the connectivity between cyber and physical systems, leaving the physical process uncontrolled and possibly drifting towards severe consequences.

Cyber attacks can cause serious security issues [61]. Under cyber attacks, e.g., by contagion of malware, security-related system features may result to be compromised and, the system safety potentially endangered. The identification of the cyber threats most affecting the system response is quite important for decision-making on optimal protection and resilience, as prevention and mitigation of malicious attacks contribute to guaranteeing CPS functionality [14, 20, 57, 64].

2.3 Hazards and Threats Diagnosis

From the perspective of integrated safety and security of CPSs, distinguishing cyber attacks from component failures is important for anticipating the potential impact on the system functionality and defining proper protection and mitigation actions for resilience.

Cyber threats aimed at altering the CPS normal operation can be diagnosed either by comparison of statistical estimates of occurrence probabilities from field data collected on the real CPS with reference values of failure probabilities of the CPS components [22, 48, 58], or by scenario processing (i.e., modeling the malicious cyber events and their manifestation on the physical domain, affecting, in turn, both cyber and physical properties of the CPS) [7, 8, 33, 51, 54].

A variety of methods for scenario processing have been proposed, based on artificial intelligence techniques. In general terms, observations are compared with the normal conditions measurements and a deviation from the legitimate data flow is found by methods such as the Sequential Probability Ratio Test (SPRT) [19, 59], the Cumulative Sum (CUSUM) chart [39, 51, 52], the Exponentially Weighted Moving Average (EWMA) inspection scheme [46], the Reversible-jump Markov Chain Monte Carlo (RJ-MCMC) [68, 72], the control charts [63] and the transfer entropy [47].

Practically, both components stochastic failures and cyber attacks occur at unknown times, leading to unpredictable changes in the distributions of physical variables that differ from the normal condition distribution. The sequential Non-Parametric CUSUM (NP-CUSUM) approach [51] has been shown capable of distinguishing normal from abnormal conditions. Based on this approach, we originally design a framework for early diagnostics of cyber attacks in CPSs. The novelty of the work lies in the reliance on the simultaneous treatment within the NP-CUSUM approach of the measurements taken from redundant channels.

3 The Advanced Lead-Cooled Fast Reactor European Demonstrator

The ALFRED reactor and the MC scheme for injecting component failures and cyber breaches are described in Sects. 3.2 and 3.3, respectively.

3.1 ALFRED Description

ALFRED is a small-size (300 MW) pool-type lead-cooled fast reactor, whose primary system configuration is shown in Fig. 1 [16]. All components of the primary cooling system, including core, primary pumps and Steam Generators (SGs), are contained in the main reactor vessel, located in a large pool within the reactor

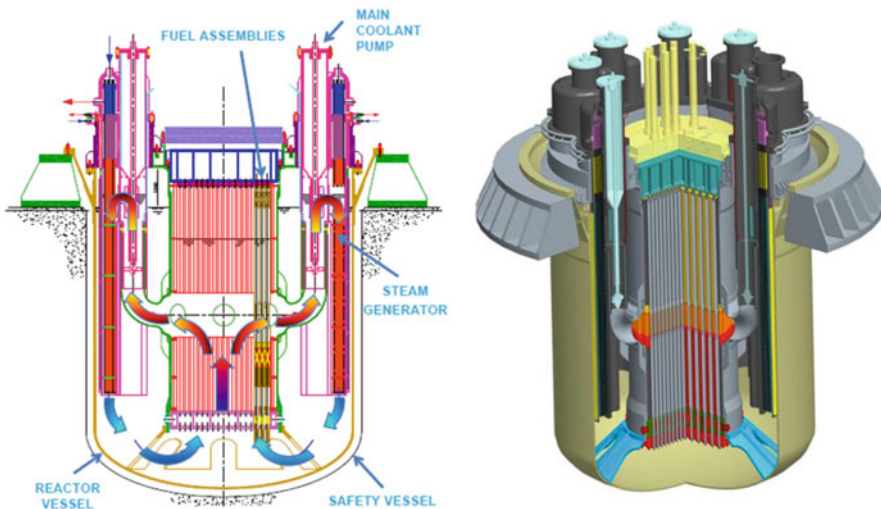


Fig. 1 ALFRED primary system layout [16]

tank. The ALFRED core providing the thermal power P_{Th} is composed by wrapped hexagonal Fuel Assemblies (FAs) with pins arranged on a triangular lattice. Control Rods (CRs) systems adjusting the heights of CRs h_{CR} have been foreseen for power regulation and reactivity swing compensation during a fuel cycle, and for scram purposes with the required reliability for a safe shutdown [17].

At full power nominal conditions, the coolant (i.e., lead) flow coming from the cold pool enters the core at temperature $T_{L,cold}$ equal to 400 °C and, once passed through the core, it is collected in the volume of the hot collector at temperature $T_{L,hot}$ equal to 480 °C; from there, it is distributed to eight parallel pipes and delivered to as many SGs. After leaving the SGs, the coolant enters the cold pool through the cold leg and returns to the core.

The eight SGs work at pressure p_{SG} equal to 180 bar. The feedwater of the secondary cooling system flows in the SGs, at pressure p_{SG} and temperature T_{feed} equal to 335 °C, and leaves the SGs after absorbing heat from the primary coolant, entering the turbine as steam at temperature T_{steam} equal to 450 °C (at full power nominal conditions). From a control point of view, it is worth noticing that the steam mass flow rate is considered proportional to the inlet pressure and governed by maneuvering the turbine valve admission (k_v), not by throttling. An attemperator is foreseen between the SG outlet header and the turbine, to limit the steam temperature at the turbine inlet T_{steam} , keeping it as close as possible to its nominal value, by adjusting the attemperator mass flow rate G_{att} .

Eventually, ALFRED produces mechanical power P_{Mech} to be transformed for the power grid.

A simplified schematics of the ALFRED primary and secondary cooling systems is shown in Fig. 2. The parameters specification of ALFRED at full power nominal conditions are reported in Table 1.

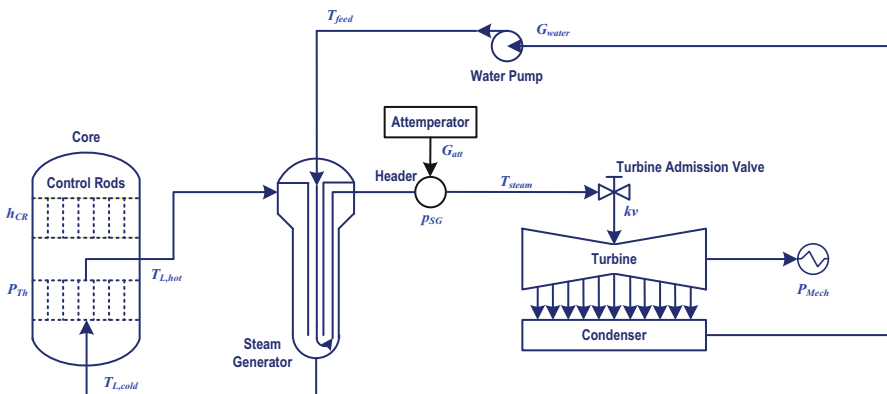


Fig. 2 ALFRED simplified schematics

Table 1 ALFRED parameters values, at full power nominal conditions

Parameter	Parameter description	Value	Unit
P_{Th}	Thermal power	$300 \cdot 10^6$	W
h_{CR}	Height of control rods	12.3	cm
$T_{L,hot}$	Coolant core outlet temperature	480	$^{\circ}\text{C}$
$T_{L,cold}$	Coolant SG outlet temperature	400	$^{\circ}\text{C}$
Γ	Coolant mass flow rate	25,984	$\text{kg} \cdot \text{s}^{-1}$
T_{feed}	Feedwater SG inlet temperature	335	$^{\circ}\text{C}$
T_{steam}	Steam SG outlet temperature	450	$^{\circ}\text{C}$
p_{SG}	SG pressure	$180 \cdot 10^5$	Pa
G_{water}	Feedwater mass flow rate	192	$\text{kg} \cdot \text{s}^{-1}$
G_{att}	Attemporator mass flow rate	0.5	$\text{kg} \cdot \text{s}^{-1}$
kv	Turbine admission valve coefficient	1	–
P_{Mech}	Mechanical power	$146 \cdot 10^6$	W

3.2 The Reactor Digital Control Scheme

A multi-loop PI (Proportional and Integral) digital control scheme, i.e., a decentralized control scheme, is developed because of its simplicity of implementation and robustness to malfunctioning of the single control loops [43]. Indeed, it can be regarded as constituted by several redundant SISO (Single Input Single Output) control loops [28].

To design the regulators and simulate the system controlled response, an object-oriented model of the entire plant has been developed (Fig. 3). Based on the Modelica language [15] and implemented in the Dymola environment [11], the corresponding simulator has been built by connecting several dedicated models for the description of the reactor (for details, see [42, 43]).

Both feedback and feedforward digital control schemes are adopted for ALFRED (see Fig. 3 shadowed part). The control aims at keeping the controlled variables of the control loops approximately at the steady state values, for operating a constant mechanical power. The PI-based feedback control configuration employs four SISO control loops independent of each other [43]. The parameters of the PI regulators reported in Table 2 are calibrated by adopting the procedures commonly employed for the SISO systems and the tuning for each PI control loop is verified by adopting the Bode criterion [28]. The values represent the optimal working conditions of the system at full power nominal conditions. The overall control scheme has been verified to effectively damp disturbances due to the change of the operating conditions. The proposed feedback scheme is improved by adding a feedforward control action, thanks to which the water mass flow rate (G_{water}) is adjusted according to the value of the thermal power (P_{Th}) exchanged at the SG interface. The implemented feedforward controller allows adjusting the heat

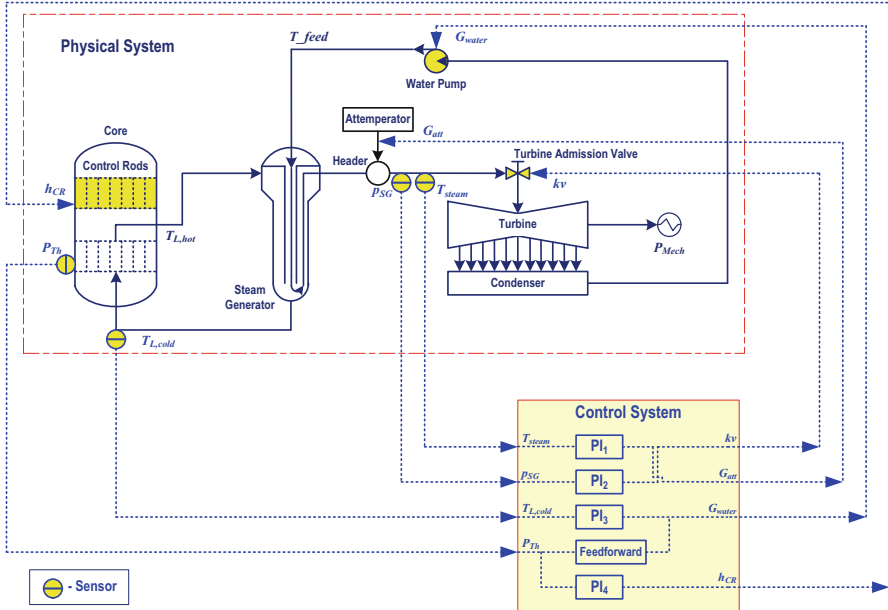


Fig. 3 ALFRED reactor control scheme

Table 2 Parameters of PI controllers

PI	Control loop		Controller parameters, $j = 1,2,3,4$	
	Controlled variable	Control variable	$K_{p,j}$	$K_{i,j}$
PI ₁	T_{steam} (°C)	G_{att} (kg·s ⁻¹)	$1 \cdot 10^{-1}$	$5 \cdot 10^{-2}$
PI ₂	P_{SG} (Pa)	k_v (-)	$3 \cdot 10^{-7}$	$1 \cdot 10^{-8}$
PI ₃	$T_{L,cold}$ (°C)	G_{water} (kg·s ⁻¹)	$6 \cdot 10^{-1}$	$1 \cdot 10^{-2}$
PI ₄	P_{Th} (W)	h_{CR} (cm)	$2 \cdot 10^{-11}$	$4 \cdot 10^{-11}$

exchange conditions in the SGs and enhancing the robustness of the control system against errors on the evaluation of the time delay between the SGs and the core due to transport phenomena.

Redundancy is commonly applied to sensors and signal processing units of a digital I&C system [3]. In the ALFRED digital control scheme, redundancy has been used to design each independent SISO loop.

Figure 4 shows an example of the redundant design scheme of the $T_{L,cold}$ -PI₃- G_{water} control loop. The real values of the coolant SG outlet temperature $T_{L,cold}(t)$ are measured by a sensor. After collected and converted to quantized (discretized) values by a data acquisition system, the measurements are duplicated by two identical digital-to-analog converters (DACs) to Subsystem 1 for computing (feeding) and 2 for monitoring, respectively. The received measurements of Subsystem 1

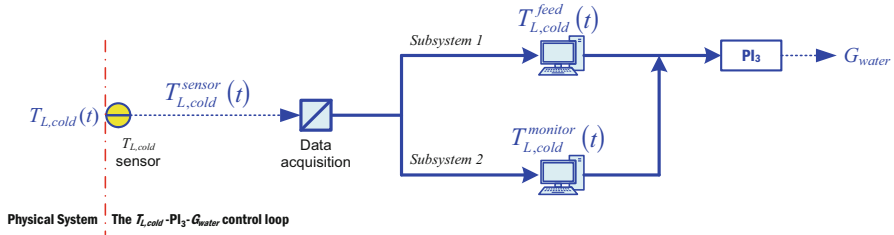


Fig. 4 The redundancy design of the $T_{L,cold}-PI_3-G_{water}$ control loop

Table 3 List of reference parameters for safety variables

Variable, y	Reference value, y^{ref} , at full power nominal conditions	Sensor measuring error $\delta_y(t)$	Converters quantization error $q_y(t)$
T_{steam} (°C)	450	$N(0,1)$	$[-0.05, +0.05]$
p_{SG} (Pa)	$180 \cdot 10^5$	$N(0,0.1) \cdot 10^5$	$[-0.01, +0.01] \cdot 10^5$
$T_{L,cold}$ (°C)	400	$N(0,1)$	$[-0.05, +0.05]$
P_{Th} (W)	$300 \cdot 10^6$	$N(0,0.5) \cdot 10^6$	$[-0.05, +0.05] \cdot 10^6$

$T_{L,cold}^{feed}(t)$ is fed to the computational unit PI_3 , whereas those of Subsystem 2 $T_{L,cold}^{monitor}(t)$ are taken as redundant data, for detecting anomalous conditions of the physical system.

Measurements are realistically considered to be affected by two types of errors [18, 60]: measurement errors (assumed distributed according to a normal distribution) and quantization errors (which are rooted in the DACs and are assumed uniformly distributed between $-1/2$ and $+1/2$ Least Significant Bit (LSB)). For simplicity, but without loss of realism, Table 3 lists the reference values of the controlled variables, the distributions of sensor measurement errors and the quantization errors that each control loops is subjected to.

In Fig. 5, measurements from the four control loops of the ALFRED are shown, on a time horizon t_M equal to 1000s,: the values of the variables are kept approximately at their nominal values, at full power nominal conditions, with some measurement errors (white noise) and quantization errors.

3.3 Failures and Cyber Breaches

To model failures and cyber attacks, a MC sampling scheme is integrated with the ALFRED model for injecting stochastic failures of sensors and cyber breaches, at random times and of random magnitudes.

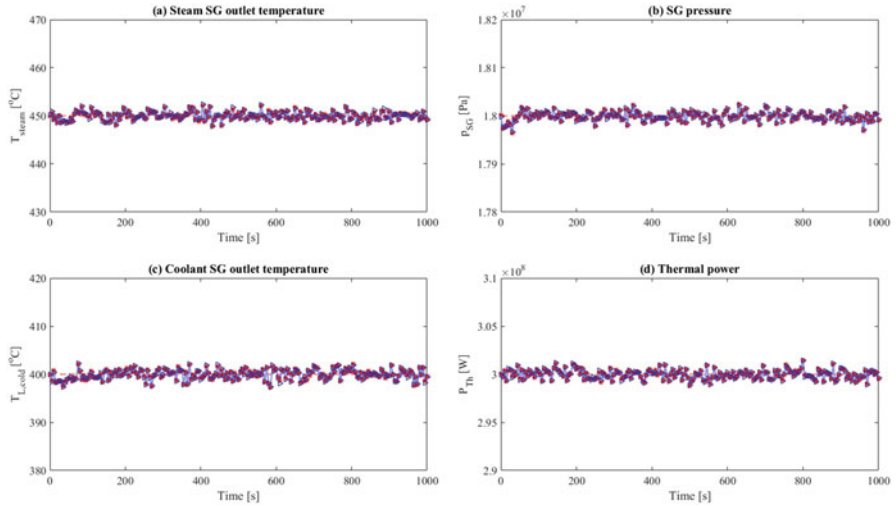


Fig. 5 Measurements from the four control loops of ALFRED at full power nominal conditions (star values for computing subsystem and triangle values for monitoring subsystem): **(a)** Steam SG outlet temperature; **(b)** SG pressure; **(c)** Coolant SG outlet temperature; and **(d)** Thermal power

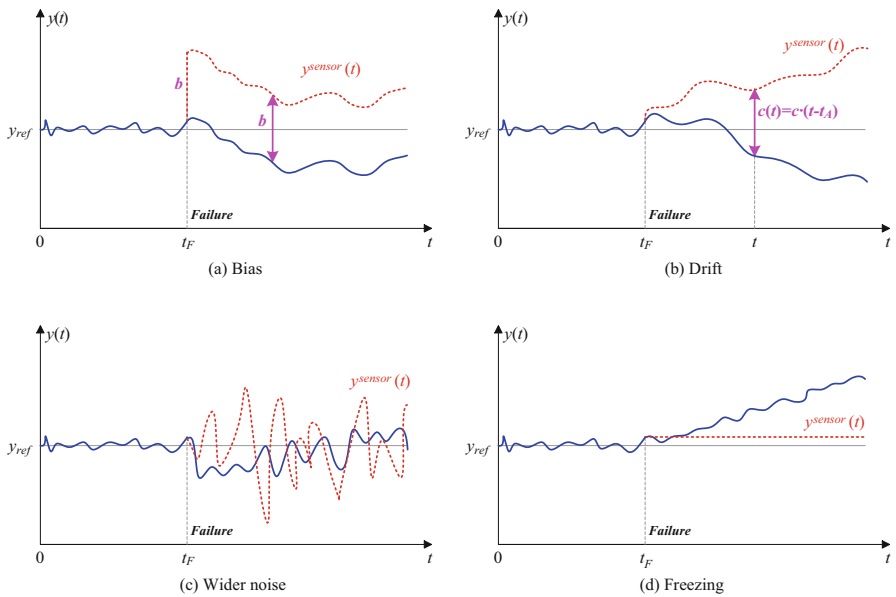


Fig. 6 Sensor failure modes: **(a)** bias; **(b)** drift; **(c)** wider noise; and **(d)** freezing. Solid lines represent the real measurements of the controlled variables, whereas dotted lines are the altered measurements of the failed sensors (for the sake of clarity, measurement and quantization errors are neglected)

Four types of sensor failure modes that may occur at random time t_R are considered [5]: (a) bias, (b) drift, (c) wider noise and (d) freezing (see dotted lines in Fig. 6a–d, respectively). The occurrence of any of these failure modes results in altered sensor measurements $y^{sensor}(t)$, as in Eq. (1):

$$y^{sensor}(t) = \begin{cases} y(t) + \delta(t), & \delta(t) = N(0, \sigma), \sigma > 0, \quad t \geq 0, \text{ normal} \\ y(t) + \delta(t) + b, & \dot{b}(t) \equiv 0, b(t_R) \neq 0, \quad t \geq t_R, \text{ bias} \\ y(t) + \delta(t) + c(t), & c(t) = c \cdot (t - t_R), \quad t \geq t_R, \text{ drift} \\ y(t) + \delta'(t), & \delta'(t) = N(0, \alpha\sigma), \alpha > 1, \quad t \geq t_R, \text{ wider noise} \\ y^{sensor}(t_R - \Delta t), & \quad t \geq t_R, \text{ freezing} \end{cases} \quad (1)$$

where $y(t)$ is the real value of the controlled variable y at time t , $\delta(t)$ is the nominal measuring error, distributed according to a normal distribution $N(0, \sigma)$, b is a constant bias factor, c is a constant drift factor, $\delta'(t)$ is a wider measuring error, distributed according to a normal distribution $N(0, \alpha\sigma)$ with a variance larger than $\delta(t)$ ($\alpha > 1$).

Without loss of generality, only the $T_{L,cold}$ (see Fig. 7) is hereafter considered (but the following discussion remains valid for any other sensor of the I&C system). Stochastic failures cause differences of the measurements $T_{L,cold}^{sensor}(t)$ from the real values of the controlled variable in the physical system. The MC sampling procedure used to inject stochastic failures to the $T_{L,cold}$ sensor at uniform random time t_R consists in sampling the uncertain parameters $b, c, \delta'(t)$ from the distributions listed in Table 4 and, then, running the ALFRED simulator for generating the controlled variables evolution throughout the mission time t_M . Erroneous measurements are, then, converted to two sets of quantized data in the data acquisition system and fed to both the computing (feeding) and monitoring subsystems.

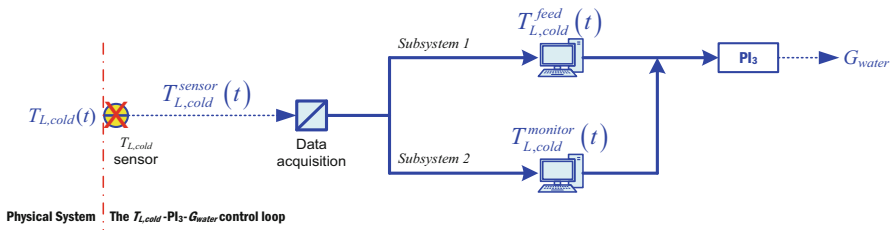


Fig. 7 Schematics of $T_{L,cold}$ sensor stochastic failures

Table 4 Parameters of sensors

Sensor	Nominal error $\delta(t)$	Failure factors			
		Bias b	Drift c	Wider noise $\delta'(t)$	Freezing
$T_{L,cold}$ (°C)	$N(0,1)$	$U(-30,30)$	$U(-1,1)$	$N(0,5)$	0

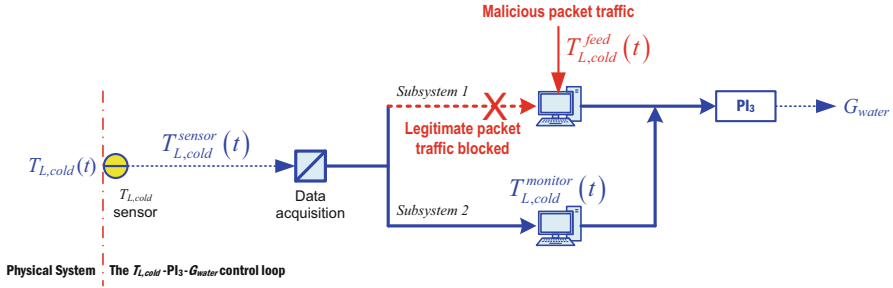


Fig. 8 Schematics of DoS attacks

Alternatively, a DoS attack is modelled to block a legitimate packet traffic that processes the genuine connection and is substituted by a malicious packet traffic [7, 51]. Figure 8 shows the schematics of a DoS attacks, in which the computing unit is fed by malicious packet traffic, altering the legitimate information, whereas, a legitimate packet traffic is regularly fed to the monitoring unit. DoS attacks are modelled to occur at uniform random time t_R within the time horizon t_M , and the uncertain parameters $b, c, \delta'(t)$ are sampled from the distributions of Table 4, as previously explained for the sensor failure.

4 The Nonparametric Cumulative Sum Approach for Real-Time Diagnostics of Cyber Attacks

The diagnostic approach is based on a NP-CUSUM algorithm of literature [51], whose details are given in Appendix A.

4.1 The Diagnostic Approach

The diagnostics approach is here illustrated with reference to the stochastic failures and the DoS attacks described in Sect. 3.3. As shown in Fig. 9, the approach involves two main steps: (i) on-line collection of measurements received by the controllers, which are fed to the NP-CUSUM algorithm that is (off-line) trained on different system behaviors to set its parameters; (ii) an on-line application of the rules of classification of failures and cyber attacks.

(i) On-line collection of measurements and application of the NP-CUSUM approach

The redundant channel measurements $Y(t)$, $Y = y^{feed}$ and $y^{monitor}$, where $y = T_{L,cold}$, are collected online by the subsystems as follows. At each time t ,

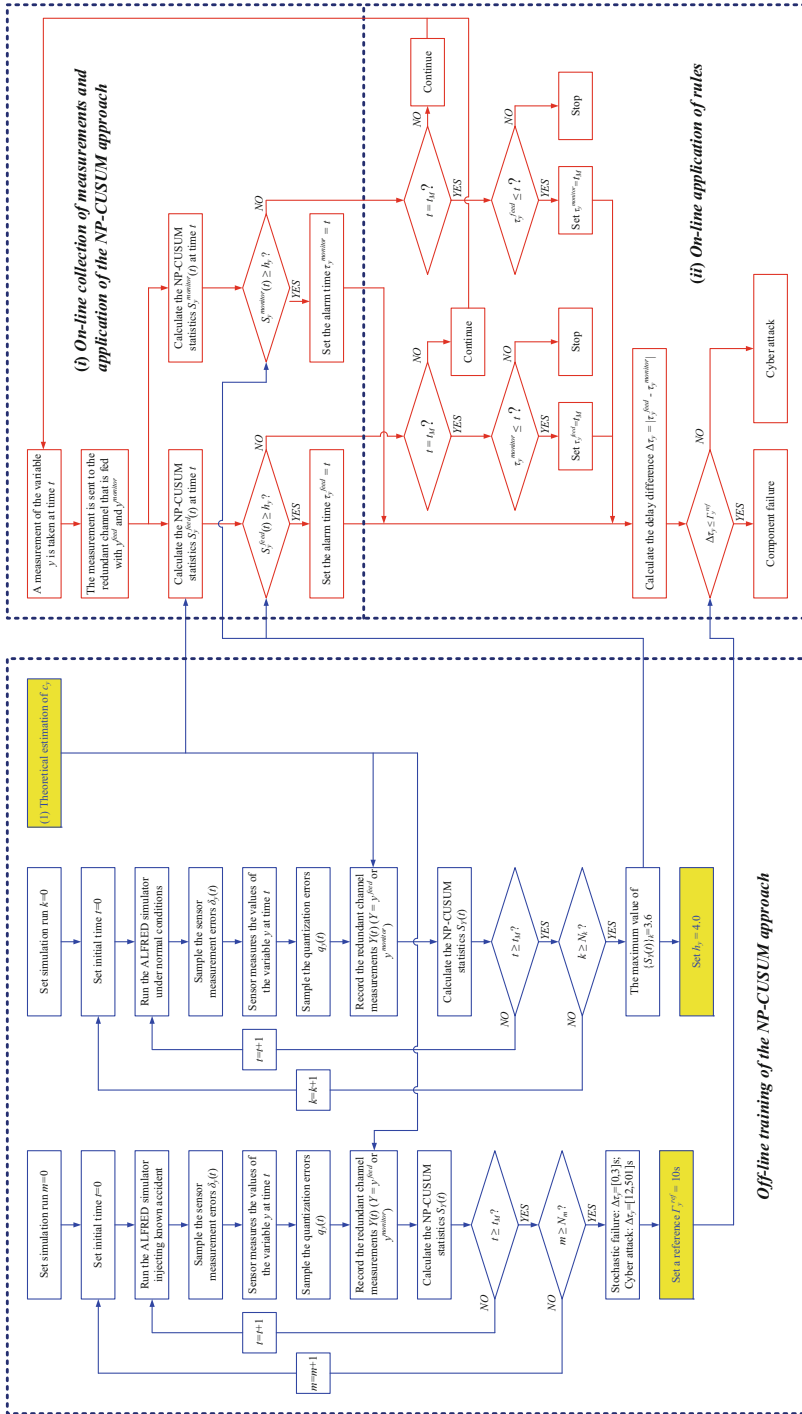


Fig. 9 Flowchart of the NP-CUSUM diagnostic approach

1. The sensor measures the values $y = T_{L,cold}$, which is affected by the sensor measurement error $\delta_y(t)$ distributed as a normal distribution of Table 3, i.e., $y^{sensor}(t) = y^{real}(t) + \delta_y(t)$;
2. The data acquisition system collects and converts $y^{sensor}(t)$ with the quantization accuracy $q_y(t)$ of Table 3, resulting in two redundant channels of quantized measurements;
3. The computing and monitoring subsystems receive the redundant measurements $Y(t)$;
4. The NP-CUSUM algorithm calculates score function-based statistics $S_Y(t)$ of the collected $Y(t)$, to check whether either $S_y^{feed}(t)$ or $S_y^{monitor}(t)$ exceeds a predefined threshold h_y :
 - If yes, record the time to alarm τ_Y (τ_y^{feed} or/and $\tau_y^{monitor}$, respectively, and proceed with the rule-based diagnostics at Step (ii));
 - If either $S_y^{feed}(t)$ or $S_y^{monitor}(t)$ exceeds h_y , collect the successive measurement because the monitored component is working under normal conditions.

(ii) **On-line application of rules**

5. If both τ_y^{feed} and $\tau_y^{monitor}$ exist, calculate the delay difference $\Delta\tau_y$ (i.e., denoting the difference between the time-to-detection delays τ_y^{feed} and $\tau_y^{monitor}$):

$$\Delta\tau_y = \left| \tau_y^{feed} - \tau_y^{monitor} \right| \quad (2)$$

Otherwise, set $\tau_y^{monitor}$ equal to t_M (when $S_y^{monitor}(t)$ has not exceeded h_y when $S_y^{feed}(t)$ does, and vice versa, respectively).

If neither exists before t_M , continue diagnostics.

6. Compare $\Delta\tau_y$ with a predefined reference delay difference Γ_y^{ref} and take decision:
 - If $\Delta\tau_y \leq \Gamma_y^{ref}$, classify the event as **Failure**;
 - If $\Delta\tau_y > \Gamma_y^{ref}$, classify the event as **Cyber Attack**.

The reference delay difference Γ_y^{ref} is estimated on a batch of $N_m = 100$ reference simulations, where, for each m -th simulation, a known component failure or cyber attack is injected. The minimum and maximum collected values of $\Delta\tau_y$ are found to be equal to 0 s and 3 s in case of components failures, and 12 s and 501 s in case of cyber attacks. Thus, we conservatively set Γ_y^{ref} equal to 10s, so that $\Delta\tau_y$ larger than 10s indicates that a cyber attack has occurred on the feeding subsystem.

(iii) Off-line training of the NP-CUSUM algorithm

The NP-CUSUM algorithm requires that the parameters c_y and h_y be customized to the different system behaviors, to guarantee the maximum capability of discriminating between failures and cyber attacks, in the ALFRED system.

(1) Estimation of c_y

A positive constant of c_y needs to be set in such a way to guarantee a negative mean value of $\mu_{\Delta g_Y} = \sum_t \Delta g_Y(Y(t)) / t$, $t = dt, 2dt, \dots, t$ ($t < t_R$), to hold before any anomaly (either failure or cyber attack) is detected, and a positive mean value $\theta_{\Delta g_Y} = \sum_t \Delta g_Y(Y(t)) / (t - t_R)$, $t = t_R, t_R + dt, t_R + 2dt, \dots$, to hold after the anomaly occurrence [51], viz:

$$\mu_{\Delta g_Y} = E[\omega_y \cdot (|Y(t) - \mu_Y| - c_y)] = -\omega_y \cdot \left(\frac{2\sigma_Y}{\sqrt{2\pi}} - c_y \right) < 0 \quad (3)$$

$$\theta_{\Delta g_Y} \geq \omega_y \cdot (|\widehat{\theta}_Y(t) - \mu_Y|_{\min} - c_y) > 0 \quad (4)$$

where, $|\widehat{\theta}_Y(t) - \mu_Y|_{\min}$ is defined as the minimum difference between the estimated post-change mean $\widehat{\theta}_{\Delta g_Y}$ and the known pre-change mean $\mu_{\Delta g_Y}$. As a result,

$$\frac{2\sigma_Y}{\sqrt{2\pi}} < c_y < |\widehat{\theta}_Y(t) - \mu_Y|_{\min} \quad (5)$$

where,

$$c_y = \varepsilon_y \cdot \widehat{\theta}_{Y,a} \quad (6)$$

where $\widehat{\theta}_{Y,a}$ is a postulated post-change mean value for an accidental scenario a .

Since under normal conditions, the probability of $Y(t)$ (distributed according to a normal distribution $N(\mu_Y, \sigma_Y)$) of falling within the interval $[\mu_Y - 2\sigma_Y, \mu_Y + 2\sigma_Y]$ is at least equal to 0.95 [10], viz:

$$\Pr[\mu_Y - 2\sigma_Y \leq Y(t) \leq \mu_Y + 2\sigma_Y] \geq 0.95 \quad (7)$$

we assume an anomaly to be occurred if $\widehat{\theta}_{Y,a}$ falls outside the interval $[\mu_Y - 2\sigma_Y, \mu_Y + 2\sigma_Y]$. Without loss of generality, we suppose that $\widehat{\theta}_{Y,a} > \mu_Y$. The minimum value of $\widehat{\theta}_{Y,a}$ results to be equal to $\mu_Y + 2\sigma_Y$ and, thus, $|\widehat{\theta}_{Y,a} - \mu_Y|_{\min}$ is equal to $2\sigma_Y$. Eqs. (5) and (6) change to:

$$\frac{1}{\sqrt{2\pi}} \left(1 - \frac{\mu_Y}{\mu_Y + 2\sigma_Y} \right) < \varepsilon_y < 1 - \frac{\mu_Y}{\mu_Y + 2\sigma_Y} \quad (8)$$

In conclusion, without loss of generality, we take a value of ε_y equal to:

$$\varepsilon_y = \frac{1}{2} \left(1 - \frac{\mu_Y}{\mu_Y + 2\sigma_Y} \right) \quad (9)$$

that, with respect to ($T_{L,cold}$ distributed as $N(400,1)^\circ\text{C}$) makes c_y turn out to be equal to 1.005°C .

(2) Estimation of h_y

The threshold h_y can be set relying on a batch of N_k reference simulations under normal conditions, whose behaviors of the variable y without change points to failures or cyber attacks can be learnt, the NP-CUSUM statistics calculated and the parameter tailored to the simulation results. Specifically, we utilize $N_k = 100$ ALFRED randomly generated simulations. For each k -th simulation,

- (a) Record the redundant channel measurements, $Y(t)$, $Y = y^{feed}$ or $y^{monitor}$, at each time t , $t = dt, 2dt, \dots, t_M$;
- (b) Calculate the corresponding NP-CUSUM statistics, $S_Y(t)$.
- (c) Set the threshold h_y such that:

$$h_y > \max_{1 \leq k \leq N_k} \{h_{y,k}\} \quad (10)$$

where,

$$h_{y,k} = \max_{1 \leq t \leq t_M} \{S_Y(t)\}_k \quad (11)$$

and, $\{S_Y(t)\}_k$ is the collection of the statistics for the k -th simulation.

As shown in Fig. 10 with respect to $T_{L,cold}$, the maximum value of the NP-CUSUM statistics is equal to 3.6 and, therefore, in what follows, we conservatively set $h_{T_{L,cold}}$ equal to 4.0.

5 Results

We illustrate the results of the NP-CUSUM-based diagnostic approach considering different $T_{L,cold}$ sensor failures and cyber attacks to the $T_{L,cold}$ -PI₃- G_{water} control loop.

5.1 Bias Failure Mode

Figure 11 presents the results of injecting bias failure at time $t_R = 630$ s with a factor b equal to 7.569°C on $T_{L,cold}$ sensor measurements. As shown in Fig. 11a, the $T_{L,cold}$ sensor bias failure deviates both measurements $T_{L,cold}^{feed}(t)$ and $T_{L,cold}^{monitor}(t)$ from the

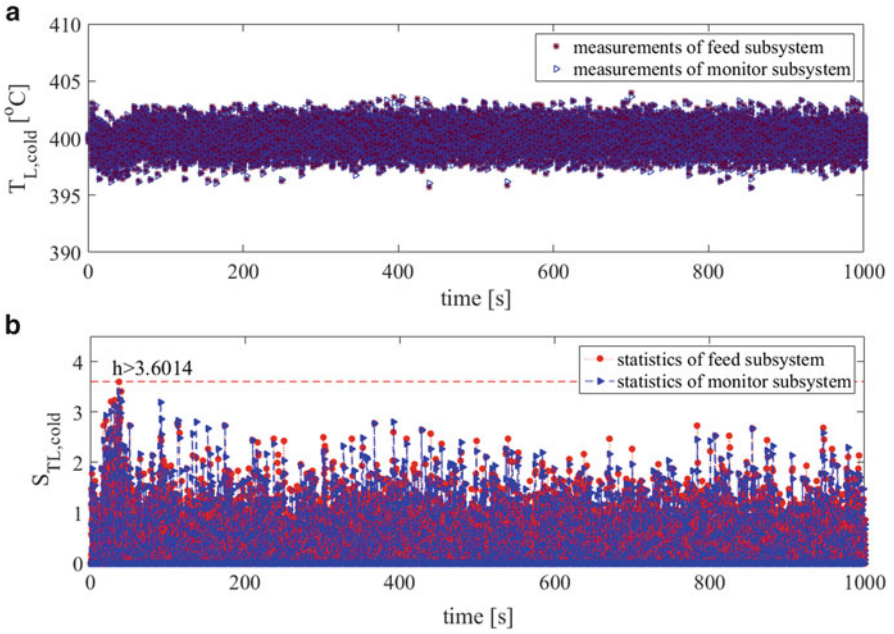


Fig. 10 Estimation of the threshold $h_{T_{L,cold}}$: (a) the received measurements of the two subsystems of the control loop; (b) the corresponding statistics calculated from the measurements

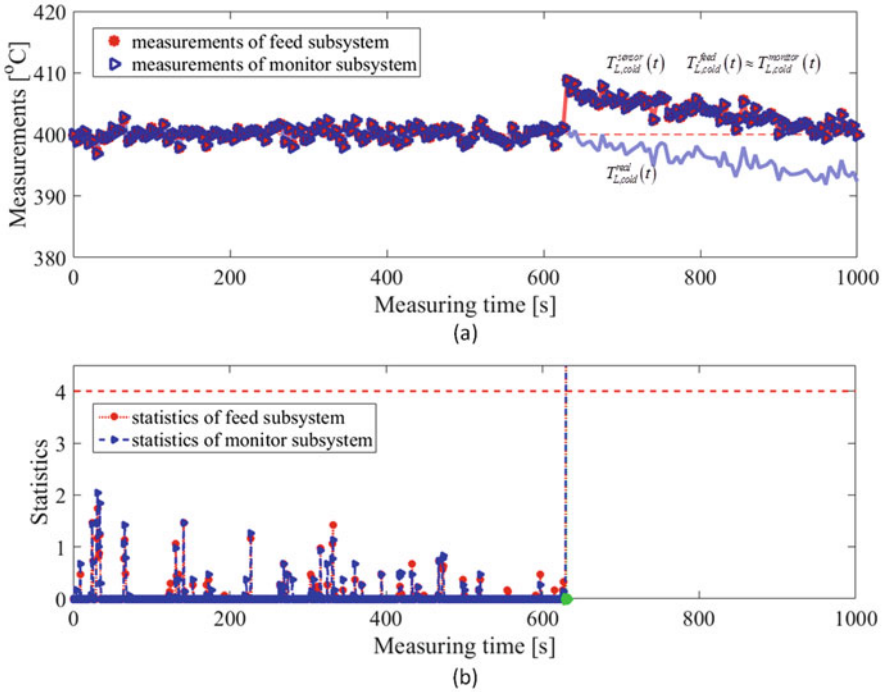


Fig. 11 $T_{L,cold}$ sensor bias failure mode: (a) the received measurements of feed and monitor Subsystems in which the bias occurs at time t_R equal to 630 s; (b) the corresponding NP-CUSUM statistics for diagnosing the bias failure

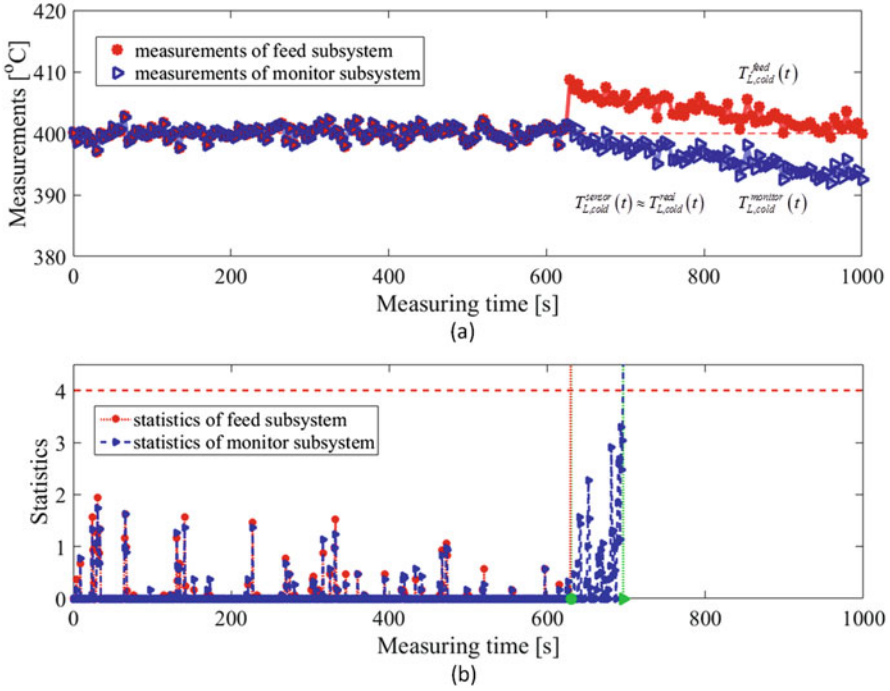


Fig. 12 Cyber attack to the computing unit mimicking a bias failure mode: (a) the received measurements of feed and monitor Subsystems in which the cyber attack occurs at time t_R equal to 630 s; (b) the corresponding NP-CUSUM statistics for diagnosing the cyber attack

real values of the physical system $T_{L,cold}^{real}(t)$. Figure 11 shows that the bias b results in very quick response of both statistics evaluated on the measurements $T_{L,cold}^{feed}(t)$ and $T_{L,cold}^{monitor}(t)$: both statistics reach quickly the threshold $h_{T_{L,cold}}$ (dotted line) and the difference $\Delta\tau_{T_{L,cold}}$ between times to alarm ($\tau_{T_{L,cold}}^{feed}$ and $\tau_{T_{L,cold}}^{monitor}$) turns out to be equal to 0 (i.e., less than Γ_y^{ref} equal to 10s) (see Fig. 11b), allowing for a (correct) identification of the event as a sensor failure mode and not as a cyber attack.

Contrarily, Fig. 12a shows a cyber attack to the computing unit mimicking a bias failure mode at $t_R = 630$ s (with b again equal to 7.569 °C): this leads $T_{L,cold}^{feed}(t)$ to deviate from $T_{L,cold}^{monitor}(t)$ (that, indeed, is the legitimate $T_{L,cold}^{sensor}(t)$ measured by the $T_{L,cold}$ sensor). The different values between the malicious and the legitimate measurements, then, lead to a delay response $\Delta\tau_{T_{L,cold}}$ equal to 66 s (larger than Γ_y^{ref}) between the threshold exceedance of $S_{T_{L,cold}}^{monitor}(t)$ and $S_{T_{L,cold}}^{feed}(t)$ (see Fig. 12b), and allowing for a (correct) identification of the event as a cyber attack.

5.2 Drift Failure Mode

Figure 13 presents the results of injecting a drift at time $t_R = 740$ s, with the drift factor c equal to 0.398. The drift c results in a very quick response of both statistics evaluated on the measurements $T_{L,cold}^{feed}(t)$ and $T_{L,cold}^{monitor}(t)$: both statistics reach quickly the threshold $h_{T_{L,cold}}$ (dotted line) and the difference $\Delta\tau_{T_{L,cold}}$ between times to alarm ($\tau_{T_{L,cold}}^{feed}$ and $\tau_{T_{L,cold}}^{monitor}$) turns out to be equal to 0 (i.e., less than Γ_y^{ref}) (see Fig. 13b), allowing for a (correct) identification of the event as a sensor failure.

Contrarily, Fig. 14a shows a cyber attack to the computing unit mimicking a drift failure mode at $t_R = 740$ s (with c again equal to 0.398), leading $T_{L,cold}^{feed}(t)$ to deviate from the legitimate $T_{L,cold}^{monitor}(t)$. The different values between the malicious and the legitimate measurements, then, lead to a delay response $\Delta\tau_{T_{L,cold}}$ equal to 41 s (larger than Γ_y^{ref}) between the threshold exceedance of $S_{T_{L,cold}}^{monitor}(t)$ and $S_{T_{L,cold}}^{feed}(t)$ (see Fig. 14b), allowing for a (correct) identification of the event as a cyber attack.

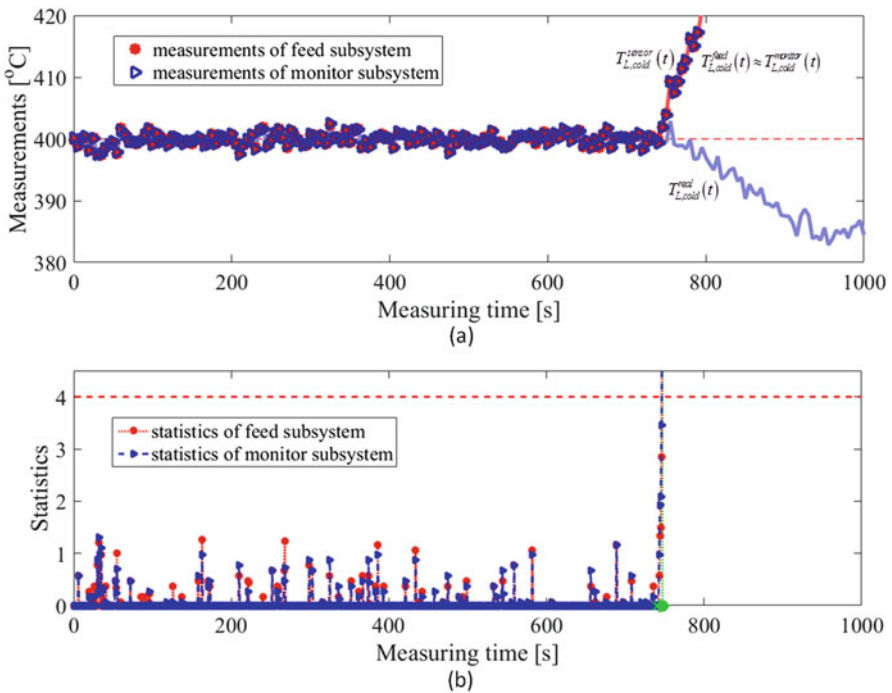


Fig. 13 $T_{L,cold}$ sensor drift failure mode: (a) the received measurements of feed and monitor Subsystems in which the drift occurs at time t_R equal to 740 s; (b) the corresponding NP-CUSUM statistics for diagnosing the bias failure

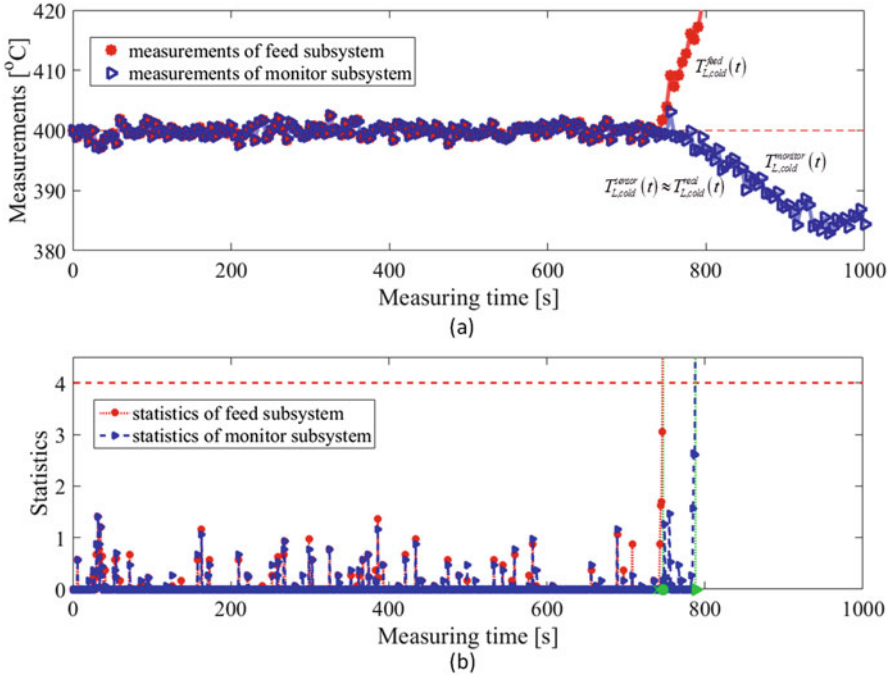


Fig. 14 Cyber attack to the computing unit mimicking a drift failure mode: (a) the received measurements of feed and monitor Subsystems in which the cyber attack occurs at time t_R equal to 740 s; (b) the corresponding NP-CUSUM statistics for diagnosing the cyber attack

5.3 Wider Noise Failure Mode

Figure 15 presents the results of injecting wider noise at time $t_R = 750$ s. This results in a very quick response of both statistics evaluated on the measurements $T_{L,cold}^{feed}(t)$ and $T_{L,cold}^{monitor}(t)$: both statistics reach quickly the threshold $h_{T_{L,cold}}$ (dotted line) and the difference $\Delta\tau_{T_{L,cold}}$ between times to alarm ($\tau_{T_{L,cold}}^{feed}$ and $\tau_{T_{L,cold}}^{monitor}$) turns out to be equal to 0 (i.e., less than Γ_y^{ref}) (see Fig. 15b), allowing for a (correct) identification of the event as a sensor failure mode.

Contrarily, Fig. 16a shows a cyber attack to the computing unit mimicking a wider noise failure mode at $t_R = 750$ s, leading $T_{L,cold}^{feed}(t)$ to deviate from the legitimate $T_{L,cold}^{monitor}(t)$. The different values between the malicious and the legitimate measurements, then, lead to a delay response $\Delta\tau_{T_{L,cold}}$ equal to 247 s (i.e., larger than Γ_y^{ref}) at t_M (see Fig. 16b), allowing for a (correct) identification of the event as a cyber attack.

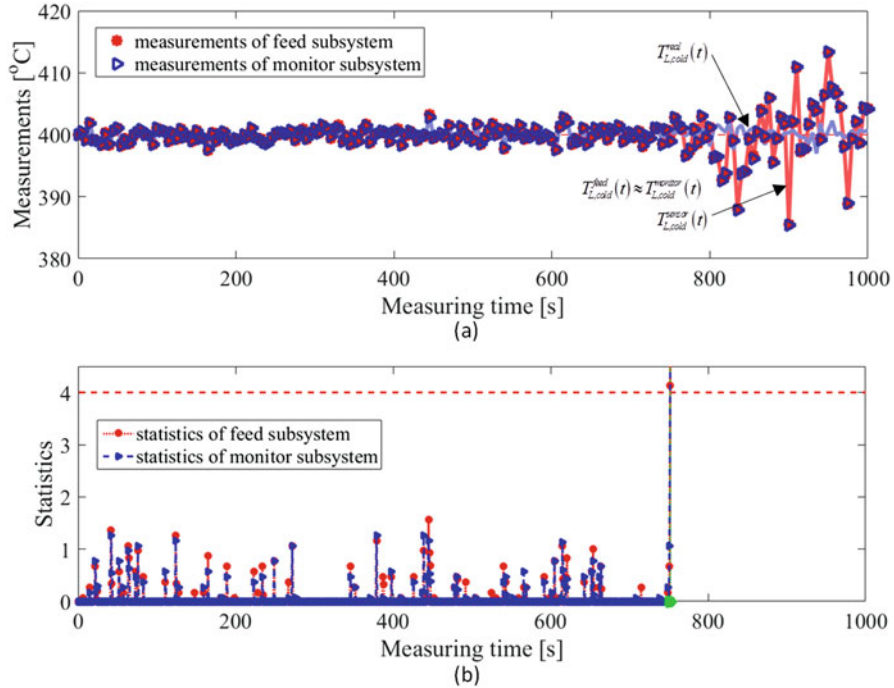


Fig. 15 $T_{L,cold}$ sensor wider noise failure mode: (a) the received measurements of feed and monitor Subsystems in which the wider noise failure occurs at time t_R equal to 750 s; (b) the corresponding NP-CUSUM statistics for diagnosing the bias failure

5.4 Freezing Failure Mode

Figure 17 presents the results of injecting freezing at time $t_R = 460$ s with the frozen $T_{L,cold}^{sensor}(t)$ equal to 402.53 °C. The freezing results in a very quick response of both statistics evaluated on the measurements $T_{L,cold}^{feed}(t)$ and $T_{L,cold}^{monitor}(t)$: Both statistics reach quickly the threshold $h_{T_{L,cold}}$ (dotted line) and the difference $\Delta\tau_{T_{L,cold}}$ between times to alarm ($\tau_{T_{L,cold}}^{feed}$ and $\tau_{T_{L,cold}}^{monitor}$) turns out to be equal to 0 (i.e., less than Γ_y^{ref}) (see Fig. 17b), allowing for a (correct) identification of the event as a sensor failure mode.

Contrarily, Fig. 18a shows a cyber attack to the computing unit mimicking a freezing failure mode at $t_R = 460$ s (with frozen $T_{L,cold}^{sensor}(t)$ again equal to 402.53 °C), leading $T_{L,cold}^{feed}(t)$ to deviate from the legitimate $T_{L,cold}^{monitor}(t)$. The different values between the malicious and the legitimate measurements, then, lead to a delay response $\Delta\tau_{T_{L,cold}}$ equal to 187 s (i.e., larger than Γ_y^{ref}) between the threshold exceedance of $S_{T_{L,cold}}^{monitor}(t)$ and $S_{T_{L,cold}}^{feed}(t)$ (see Fig. 18b), allowing for a (correct) identification of the event as a cyber attack.

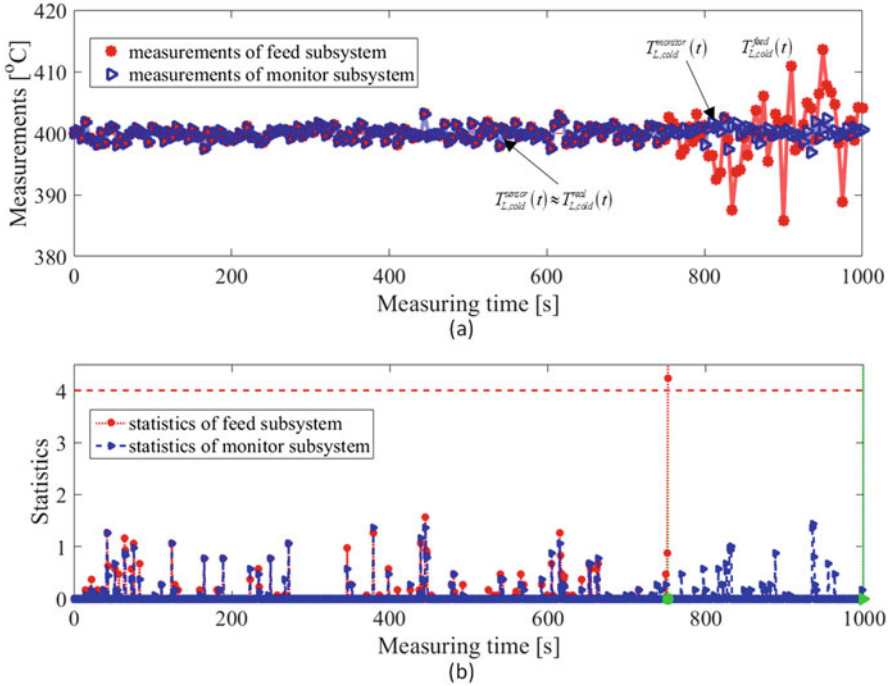


Fig. 16 Cyber attack to the computing unit mimicking a wider noise failure mode: (a) the received measurements of feed and monitor Subsystems in which the cyber attack occurs at time t_R equal to 750 s; (b) the corresponding NP-CUSUM statistics for diagnosing the cyber attack

The results of these illustrative examples show that the NP-CUSUM-based diagnostics approach is capable of diagnosing cyber attacks, distinguishing them from stochastic failures of components, based on the identified rules of assignments.

6 Performance of the Diagnostic Approach

The previous examples shown in Sect. 5 demonstrate the effectiveness of the NP-CUSUM diagnostics approach. Since the proposed diagnostic approach may suffer from either large false alarm rate (if the threshold is set too small) or high missed alarm rate (if the threshold is set too large) [9], an extensive and massive test with respect to unknown sensor failures and/or unknown cyber attacks is performed for assessing its diagnostic capabilities. We calculate false alarm, missed alarm and misclassification rates with respect to 100 randomly sampled stochastic failures and 100 different cyber attacks for each failure mode (i.e., bias, drift, wider noise or freezing) (thus, a total of $N_A = 800$ runs). At each run of the simulation: a random time t_R within the mission time $t_M = 1000$ s and an uncertain parameter value (i.e.,

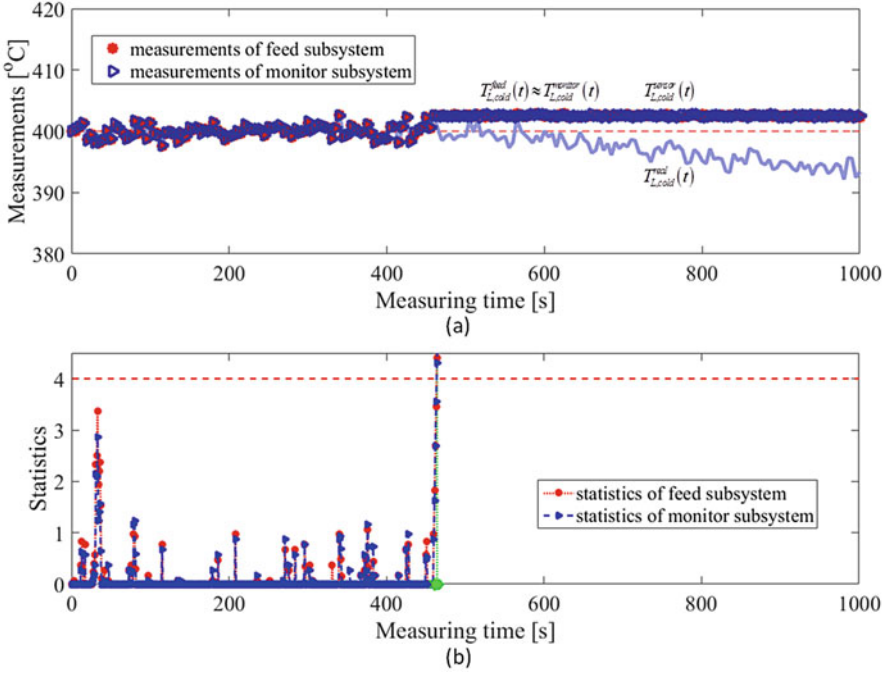


Fig. 17 $T_{L,cold}$ sensor freezing failure mode: (a) the received measurements of feed and monitor in which the freezing occurs at time t_R equal to 460 s; (b) the corresponding NP-CUSUM statistics for diagnosing the bias failure

b for bias, c for drift, $\delta'(t)$ are sampled from the distributions listed in Table 4 for wider noise or frozen value for freezing) and used to inject a $T_{L,cold}$ sensor failure or a cyber attack to the computing unit. Then, the NP-CUSUM-based diagnostic algorithm is applied to both $T_{L,cold}^{feed}(t)$ and $T_{L,cold}^{monitor}(t)$, to calculate $S_{L,cold}^{feed}(t)$ and $S_{L,cold}^{monitor}(t)$, respectively. The diagnostic performances are measured as follows:

- False alarm rate $\alpha_{T_{L,cold}}^h$: the probability of either $S_{L,cold}^{feed}(t)$ or $S_{L,cold}^{monitor}(t)$ in an accidental scenario exceeding the threshold $h_{T_{L,cold}}$ before t_R .
- Missed alarm rate $\beta_{T_{L,cold}}^h$: the probability of neither $S_{L,cold}^{feed}(t)$ nor $S_{L,cold}^{monitor}(t)$ in an accidental scenario exceeding the threshold $h_{T_{L,cold}}$ within the mission time t_M .
- Misclassification rate $\gamma(\Gamma_{T_{L,cold}}^{ref})$: given a reference delay difference $\Gamma_{T_{L,cold}}^{ref}$, the probability of a misclassified assignment of an event.

Table 5 lists the estimates of $\alpha_{T_{L,cold}}^h$ and $\beta_{T_{L,cold}}^h$ with respect to the threshold $h_{T_{L,cold}}$ equal to 4.0, among the total of $N_A = 800$ runs of stochastic failures and cyber attacks. The results in the Table show that the total values of $\alpha_{T_{L,cold}}^h$ and

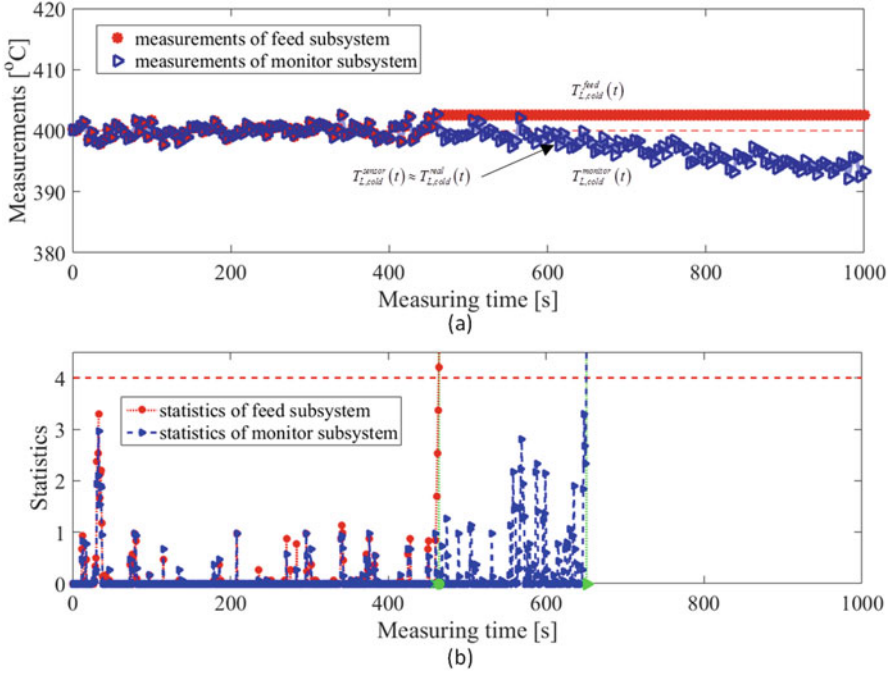


Fig. 18 Cyber attack to the computing unit mimicking a freezing failure mode: (a) the received measurements of feed and monitor Subsystems in which the cyber attack occurs at time t_R equal to 460 s; (b) the corresponding NP-CUSUM statistics for diagnosing the cyber attack

Table 5 False and missed alarm rates with respect to $h_{T_{L,cold}}$

Character	Bias	Drift	Wider noise	Freezing	Total
$\alpha_{T_{L,cold}}^h$	8/200	8/200	1/200	8/200	25/800 = 0.0313
$\beta_{T_{L,cold}}^h$	13/200	5/200	0/200	2/200	20/800 = 0.0250

$\beta_{T_{L,cold}}^h$ are equal to 0.0313 and 0.0250, respectively, and the low values are accepted in the diagnostics of cyber attacks of the ALFRED.

To analyze the effect of an improper choice of $\Gamma_{T_{L,cold}}^{ref}$ that may mistakenly ascribe an accidental scenario to inconsistent reasons and lead to misclassified diagnostics, we estimate $\gamma \left(\Gamma_{T_{L,cold}}^{ref} \right)$ among the $N_A = 800$ scenarios, with respect to different values of $\Gamma_{T_{L,cold}}^{ref}$. We assess the misclassification rates by defining four misclassification types (i.e., Misclassification I, II, III and IV), that differ in terms of the difference between alarm delays $\Delta \tau_{T_{L,cold}}$ to a reference value $\Gamma_{T_{L,cold}}^{ref}$ in Table 6.

Figure 19 shows the calculated misclassification rates $\gamma \left(\Gamma_{T_{L,cold}}^{ref} \right)$ varying with $\Gamma_{T_{L,cold}}^{ref}$ from 0 to 60. $\gamma \left(\Gamma_{T_{L,cold}}^{ref} \right)$ is calculated by summing all the misclassified assignments of the accidental scenarios, which are recorded in the way of false and

Table 6 Misclassification assessment with respect to $\Gamma_{T_{L,cold}}^{ref}$

Real scenario	Comparison	Assignment	Check	False alarm of	Missed alarm of
Sensor failure	$\Delta\tau_{T_{L,cold}} \leq \Gamma_{T_{L,cold}}^{ref}$	Sensor failure	Correct	-	-
	$\Delta\tau_{T_{L,cold}} > \Gamma_{T_{L,cold}}^{ref}$	Cyber attack	Misclassification I	Cyber attack	Sensor failure
	Neither $\tau_{T_{L,cold}}^{feed}$ nor $\tau_{T_{L,cold}}^{monitor}$	Normal condition	Misclassification II	-	Sensor failure
Cyber attack	$\Delta\tau_{T_{L,cold}} \leq \Gamma_{T_{L,cold}}^{ref}$	Sensor failure	Misclassification III	Sensor failure	Cyber attack
	$\Delta\tau_{T_{L,cold}} > \Gamma_{T_{L,cold}}^{ref}$	Cyber attack	Correct	-	-
	Neither $\tau_{T_{L,cold}}^{feed}$ nor $\tau_{T_{L,cold}}^{monitor}$	Normal condition	Misclassification IV	-	Cyber attack

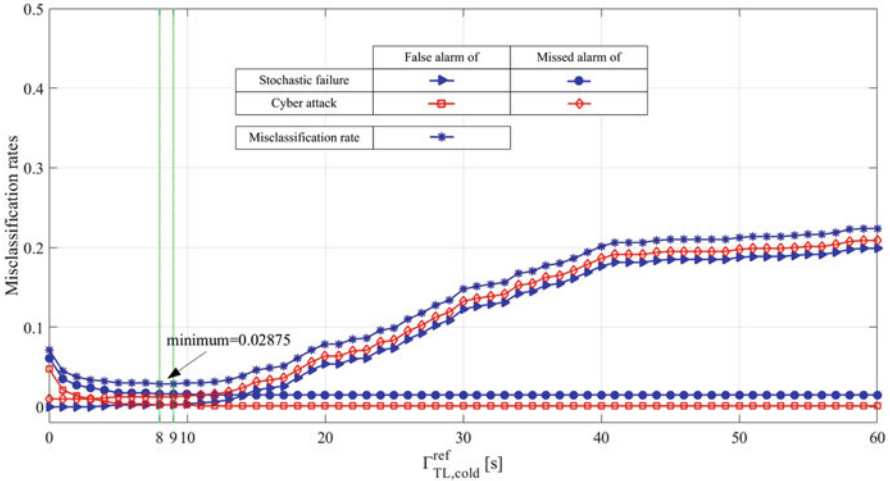


Fig. 19 The misclassification rates varying with $\Gamma_{TL,cold}^{ref}$

missed alarm of sensor failures and of cyber attacks, respectively. Results show that the minimum misclassification rate (equal to 0.02875) can be achieved if the categorical difference $\Gamma_{TL,cold}^{ref}$ is optimally equal to 8 s or 9 s. It is also noted that, the minimum rate being larger than $\beta_{TL,cold}^h$ (equal to 0.025) turns out to be reasonable because the identified misclassification scenarios here include the missed alarms identified with respect to $h_{TL,cold}$ equal to 4.0.

7 Conclusions

In this chapter, we presented a nonparametric cumulative sum (NP-CUSUM) approach to enable real-time diagnosis of cyber attacks on Cyber-Physical Systems (CPSs). The diagnostics approach allows distinguishing between components failures and cyber attacks to the controllers, guiding decisions for recovering CPSs from anomalies.

The diagnostic performance of the approach has been analyzed by the false and missed alarm rates, with reference to a prespecified threshold, and the misclassification rates varying with the reference delay differences for identifying a cyber attack or a sensor failure.

We have applied the diagnostics approach to the digital Instrumentation and Control (I&C) system of the Advanced Lead-cooled Fast Reactor European Demonstrator (ALFRED). Cyber breach events attacking the embedded CPS controllers and sensor failures are injected by a Monte Carlo sampling procedure, at random times and with random magnitudes. Results show that the diagnostic approach is

capable of identifying most of the generated failure/attack scenarios, with low false alarm rate, missed alarm rate and misclassification rate.

Future work will regard, on one hand, the optimization of the threshold setting and the decision of the reference delay difference, for further minimizing the false and missed alarms, and, on the other hand, the development of an extended multi-variable/channel NP-CUSUM diagnostics framework (e.g., for all the four control loops in the digital I&C system of ALFRED), for localizing and recognizing the failures and/or cyber attacks.

Acknowledgement The authors are thankful to Prof. Antonio Cammi and Dr. Stefano Lorenzi of the Energy Department, Politecnico di Milano, for providing guidance and training on code simulating the ALFRED reactor.

Appendix A: The NP-CUSUM Algorithm

Without loss of generality, let us consider an accidental scenario a simulated over a mission time t_M , during which a cyber attack occurs at random time t_R ($t_R < t_M$). Considering a time interval dt , we can define the pre-attack signal mean value $\mu_Y(Y(t)) = \sum_t Y(t)/t, t = dt, 2dt, \dots, t, (t < t_R)$, where $Y(t)$ is the measurement Y of a controlled variable y at time t under normal operation conditions (see Fig. A.1a, for example). Assume that DoS attacks lead to arbitrary and abrupt changes in the distributions of observations, such that the (unknown) post-attack mean value results to be $\theta_Y(Y(t)) = \sum_t Y(t)/(t - t_R), t = t_R, t_R + dt, t_R + 2dt, \dots$.

We define a score function $g_Y(Y(t))$ as:

$$g_Y(Y(t)) = \sum_t \omega_y \cdot \Lambda(Y(t)) = \sum_t \omega_y \cdot (|Y(t) - \mu_Y| - c_y(t)) \tag{A.1}$$

where ω_y is a positive weight that is used for normalizing $\Lambda(Y(t))$ and chosen equal to $1/\sigma_Y$, where σ_Y is the standard deviation of $Y(t), t = dt, 2dt, \dots$, and the parameter $c_y(t)$ depends on the past $t-1$ measurements as in Eq. (A.2):

$$c_y(t) = \varepsilon_y \cdot \widehat{\theta}_Y(t) \tag{A.2}$$

where ε_y is a tuning parameter belonging to the interval (0,1) and $\widehat{\theta}_Y(t)$ is an estimate of the unknown mean value $\theta_Y(Y(t))$. In practice, it is difficult to estimate $\widehat{\theta}_Y(t)$ on-line. Hence, Eq. (A.1) is simplified in:

$$\Delta g_Y(Y(t)) = \omega_y \cdot (|Y(t) - \mu_Y| - c_y) \tag{A.3}$$

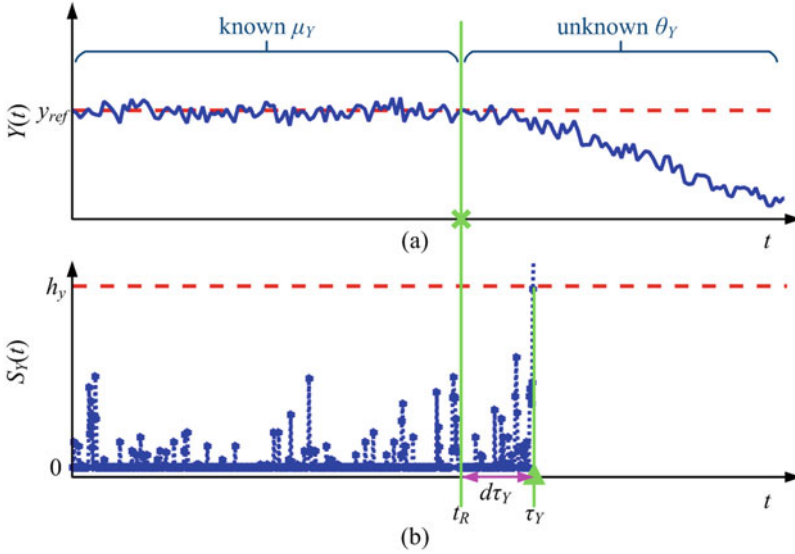


Fig. A.1 The NP-CUSUM algorithm: **(a)** a stream of measurement $Y(t)$ of an accidental scenario in which a cyber attack occurring at time t_R ; **(b)** the corresponding NP-CUSUM statistic $S_Y(t)$ for diagnosing the cyber attack at the time to alarm τ_Y

The score function $S_Y(t)$ adopted in the NP-CUSUM algorithm is, then, defined as:

$$S_Y(t) = \max \{0, S_Y(t-1) + \Delta g_Y(Y(t))\} \quad (\text{A.4})$$

where, $S_Y(0) = 0$.

In practice, with respect to a stream of measurement $Y(t)$, the NP-CUSUM statistics $S_Y(t)$ remain close to zero or slightly positive under normal operation conditions, whereas, it starts drifting and increasing when a cyber attack occurs at time t_R and, ends up with exceeding a predefined positive threshold h_y (see Fig. A.1b). An alarm can be triggered when $S_Y(t)$ reaches h_y at the time of alarm:

$$\tau_Y = \min \{t \geq 1 : S_Y(t) \geq h_y\} \quad (\text{A.5})$$

The detection delay dt_Y between t_R and τ_Y depends on the choice of h_y . A good diagnostic algorithm is expected to perform with a low False Alarm Rate (FAR) and a small value dt_Y .

References

1. Aldemir T, Guarro S, Mandelli D, Kirschenbaum J, Mangan LA, Bucci P et al (2010) Probabilistic risk assessment modeling of digital instrumentation and control systems using two dynamic methodologies. *Reliab Eng Syst Saf* 95(10):1011–1039
2. Alur R (2015) Principles of cyber-physical systems. MIT Press, Cambridge, MA
3. Authen S, Holmberg JE (2012) Reliability analysis of digital systems in a probabilistic risk analysis for nuclear power plants. *Nucl Eng Technol* 44(5):471–482
4. Aven T (2009) Identification of safety and security critical systems and activities. *Reliab Eng Syst Saf* 94(2):404–411
5. Boskvic JD, Mehra RK (2002) Stable adaptive multiple model-based control design for accommodation of sensor failures. In: American control conference, 2002. Proceedings of the 2002, IEEE, vol 3, pp 2046–2051
6. Bradley JM, Atkins EM (2015) Optimization and control of cyber-physical vehicle systems. *Sensors* 15(9):23020–23049
7. Carl G, Kesidis G, Brooks RR, Rai S (2006) Denial-of-service attack-detection techniques. *IEEE Internet Comput* 10(1):82–89
8. Debar H, Dacier M, Wespi A (1999) Towards a taxonomy of intrusion-detection systems. *Comput Netw* 31(8):805–822
9. Di Maio F, Baraldi P, Zio E, Seraoui R (2013) Fault detection in nuclear power plants components by a combination of statistical methods. *IEEE Trans Reliab* 62(4):833–845
10. Duda RO, Hart PE, Stork DG (1973) Pattern classification, vol 2. Wiley, New York, pp 526–528
11. DYMOLA (2015) Dymola (Version 2015). France: Dassault Systèmes. Retrieved from <http://www.3ds.com/products-services/catia/products/dymola>
12. Eames DP, Moffett J (1999) The integration of safety and security requirements. In: International conference on computer safety, reliability, and security. Springer, Berlin/Heidelberg, pp 468–480
13. Elhag S, Fernández A, Bawakid A, Alshomrani S, Herrera F (2015) On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on intrusion detection systems. *Expert Syst Appl* 42(1):193–202
14. Fang Y, Sansavini G (2017) Optimizing power system investments and resilience against attacks. *Reliab Eng Syst Saf* 159:161–173
15. Fritzson P (2010) Principles of object-oriented modeling and simulation with Modelica 2.1. Wiley, Hoboken
16. Frogheri M, Alemberti A, Mansani L (2015) The lead fast reactor: demonstrator (ALFRED) and ELFR design. In: Fast reactors and related fuel cycles: safe technologies and sustainable scenarios (FR13). V. 1. Proceedings of an international conference
17. Grasso G, Petrovich C, Mikityuk K, Mattioli D, Manni F, Gugiu D (2013) Demonstrating the effectiveness of the European LFR concept: the ALFRED core design. In: Proceedings of the IAEA international conference on fast reactors and related fuel cycles: safe technologies and sustainable scenarios
18. Gray R, Neuhoff D (1998) Quantization. *IEEE Trans Inf Theory* 44(6):2325–2383
19. Hines JW, Garvey DR (2006) Development and application of fault detectability performance metrics for instrument calibration verification and anomaly detection. *J Pattern Recogn Res* 1(1):2–15
20. Hu X, Xu M, Xu S, Zhao P (2017) Multiple cyber attacks against a target with observation errors and dependent outcomes: characterization and optimization. *Reliab Eng Syst Saf* 159:119–133
21. IAEA (2009) Implementing digital instrumentation and control systems in the modernization of nuclear power plants. Technical report NP-T-1.4. IAEA

22. Jockenhövel-Bartfeld M, Taurines A, Hessler C (2016) Quantification of application software failures of digital I&C in probabilistic safety analyses. In: 13th international conference on probabilistic safety assessment and management, Seoul, Korea
23. Khaitan SK, McCalley JD (2015) Design techniques and applications of cyberphysical systems: a survey. *IEEE Syst J* 9(2):350–365
24. Kim KD, Kumar PR (2012) Cyber–physical systems: a perspective at the centennial. *Proc IEEE* 100(Special Centennial Issue):1287–1308
25. Kornecki AJ, Liu M (2013) Fault tree analysis for safety/security verification in aviation software. *Electronics* 2(1):41–56
26. Kriaa S, Pietre-Cambacèdes L, Bouissou M, Halgand Y (2015) A survey of approaches combining safety and security for industrial control systems. *Reliab Eng Syst Saf* 139:156–178
27. Lee EA (2008) Cyber physical systems: design challenges. In: Object oriented real-time distributed computing (ISORC), 2008 11th IEEE international symposium on, IEEE, pp 363–369
28. Levine WS (ed) (1996) *The control handbook*. CRC Press, Boca Raton
29. Li J, Huang X (2016) Cyber attack detection of I&C systems in NPPS based on physical process data. In: 2016 24th international conference on nuclear engineering, American Society of Mechanical Engineers, pp V002T07A011–V002T07A011
30. Liang G, Zhao J, Luo F, Weller SR, Dong ZY (2017) A review of false data injection attacks against modern power systems. *IEEE Trans Smart Grid* 8(4):1630–1638
31. Machado, R. C., Boccardo, D. R., De Sá, V. G. P., & Szwarcfiter, J. L. (2016). Software control and intellectual property protection in cyber-physical systems. *EURASIP J Inf Secur*, 2016(1), 8
32. McNelles P, Zeng ZC, Renganathan G, Lamarre G, Akl Y, Lu L (2016) A comparison of fault trees and the dynamic flowgraph methodology for the analysis of FPGA-based safety systems part 1: reactor trip logic loop reliability analysis. *Reliab Eng Syst Saf* 153:135–150
33. Mo Y, Chabukswar R, Sinopoli B (2014) Detecting integrity attacks on SCADA systems. *IEEE Trans Control Syst Technol* 22(4):1396–1407
34. Mohammadpourfard M, Sami A, Seifi AR (2017) A statistical unsupervised method against false data injection attacks: a visualization-based approach. *Expert Syst Appl* 84:242–261
35. Moteff JD (2012) *Critical infrastructure resilience: the evolution of policy and programs and issues for congress*. Congressional Research Service, Library of Congress, Washington, DC
36. Ntalampiras S (2015) Detection of integrity attacks in cyber-physical critical infrastructures using ensemble modeling. *IEEE Trans Ind Inf* 11(1):104–111
37. Ntalampiras S (2016) Automatic identification of integrity attacks in cyber-physical systems. *Expert Syst Appl* 58:164–173
38. Obama B (2013) *Presidential policy directive 21: critical infrastructure security and resilience*. The White House, Washington, DC
39. Page ES (1954) Continuous inspection schemes. *Biometrika* 41(1/2):100–115
40. Pajic M, Weimer J, Bezzo N, Sokolsky O, Pappas GJ, Lee I (2017) Design and implementation of attack-resilient cyberphysical systems: with a focus on attack-resilient state estimators. *IEEE Control Syst* 37(2):66–81
41. Piètre-Cambacèdes L, Bouissou M (2013) Cross-fertilization between safety and security engineering. *Reliab Eng Syst Saf* 110:110–126
42. Ponciroli R, Bigoni A, Cammi A, Lorenzi S, Luzzi L (2014) Object-oriented modelling and simulation for the ALFRED dynamics. *Prog Nucl Energy* 71:15–29
43. Ponciroli R, Cammi A, Della Bona A, Lorenzi S, Luzzi L (2015) Development of the ALFRED reactor full power mode control system. *Prog Nucl Energy* 85:428–440
44. Qiu P, Hawkins D (2003) A nonparametric multivariate cumulative sum procedure for detecting shifts in all directions. *J R Stat Soc Ser D Stat* 52(2):151–164
45. Rahman MS, Mahmud MA, Oo AM, Pota HR (2017) Multi-agent approach for enhancing security of protection schemes in cyber-physical energy systems. *IEEE Trans Ind Inf* 13(2):436–447

46. Roberts SW (1959) Control chart tests based on geometric moving averages. *Technometrics* 1(3):239–250
47. Shi D, Guo Z, Johansson KH, Shi L (2018) Causality countermeasures for anomaly detection in cyber-physical systems. *IEEE Trans Autom Control* 63(2):386–401
48. Shin J, Son H, Heo G (2015) Development of a cyber security risk model using Bayesian networks. *Reliab Eng Syst Saf* 134:208–217
49. Skogestad S, Postlethwaite I (2007) *Multivariable feedback control: analysis and design*, vol 2. Wiley, New York, pp 359–368
50. Tan R, Nguyen HH, Foo EY, Yau DK, Kalbarczyk Z, Iyer RK, Gooi HB (2017) Modeling and mitigating impact of false data injection attacks on automatic generation control. *IEEE Trans Inf Forensics Secur* 12(7):1609–1624
51. Tartakovsky AG, Rozovskii BL, Blažek RB, Kim H (2006a) Detection of intrusions in information systems by sequential change-point methods. *Stat Methodol* 3(3):252–293
52. Tartakovsky AG, Rozovskii BL, Blazek RB, Kim H (2006b) A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods. *IEEE Trans Signal Process* 54(9):3372–3382
53. Tartakovsky AG, Polunchenko AS, Sokolov G (2013) Efficient computer network anomaly detection by changepoint detection methods. *IEEE J Sel Top Sign Proces* 7(1):4–11
54. Teixeira A, Amin S, Sandberg H, Johansson KH, Sastry SS (2010) Cyber security analysis of state estimators in electric power systems. In: *Decision and control (CDC), 2010 49th IEEE conference on*, IEEE, pp 5991–5998
55. Trabelsi Z, Rahmani H (2005) An anti-sniffer based on ARP cache poisoning attack. *Inf Syst Secur* 13(6):23–36
56. Wang W, Di Maio F, Zio E (2016) Component-and system-level degradation modeling of digital instrumentation and control systems based on a multi-state physics modeling approach. *Ann Nucl Energy* 95:135–147
57. Wang W, Cammi A, Di Maio F, Lorenzi S, Zio E (2017a) A Monte Carlo-based exploration framework for identifying components vulnerable to cyber threats in nuclear power plants. *Reliab Eng Syst Saf* 175:24–37
58. Wang W, Di Maio F, Zio E (2017b) Estimation of failure on-demand probability and malfunction rate values in cyber-physical systems of nuclear power plants. In: *The 2017 international topical meeting on probabilistic safety assessment and analysis (PSA2017)*, Pittsburgh, USA, September, 2017, pp 24–28
59. Wald A (1973) *Sequential analysis*. Courier Corporation, New York
60. Widrow B (1961) Analysis of amplitude-quantized sampled-data systems. *Electr Eng* 80(6):450–450
61. Xiang Y, Wang L, Liu N (2017) Coordinated attacks on electric power systems in a cyber-physical environment. *Electr Power Syst Res* 149:156–168
62. Xie M, Goh TN, Ranjan P (2002) Some effective control chart procedures for reliability monitoring. *Reliab Eng Syst Saf* 77(2):143–150
63. Yuan Y, Zhu Q, Sun F, Wang Q, Başar T (2013) Resilient control of cyber-physical systems against denial-of-service attacks. In: *Resilient control systems (ISRCS), 2013 6th international symposium on*, IEEE, pp 54–59
64. Yuan W, Zhao L, Zeng B (2014) Optimal power grid protection through a defender-attacker-defender model. *Reliab Eng Syst Saf* 121:83–89
65. Zalewski J, Buckley IA, Czejdo B, Drager S, Kornecki AJ, Subramanian N (2016) A framework for measuring security as a system property in cyberphysical systems. *Information* 7(2):33
66. Zargar ST, Joshi J, Tipper D (2013) A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. *IEEE Commun Surv Tutor* 15(4):2046–2069
67. Zaytoon J, Lafortune S (2013) Overview of fault diagnosis methods for discrete event systems. *Annu Rev Control* 37(2):308–320
68. Zhao X, Chu PS (2010) Bayesian changepoint analysis for extreme events (typhoons, heavy rainfall, and heat waves): an RJMCMC approach. *J Clim* 23(5):1034–1046

69. Zio E (2009) Reliability engineering: old problems and new challenges. *Reliab Eng Syst Saf* 94(2):125–141
70. Zio E (2016) Challenges in the vulnerability and risk analysis of critical infrastructures. *Reliab Eng Syst Saf* 152:137–150
71. Zio E, Di Maio F (2009) Processing dynamic scenarios from a reliability analysis of a nuclear power plant digital instrumentation and control system. *Ann Nucl Energy* 36(9):1386–1399
72. Zio E, Zoia A (2009) Parameter identification in degradation modeling by reversible-jump Markov Chain Monte Carlo. *IEEE Trans Reliab* 58(1):123–131

Index

A

Arrighi, Chiara, 177

C

Carnevali, Laura, 177

Castelli, Fabio, 177

D

Debertol, Daniele, 49

Di Maio, Francesco, 195

F

Ferrari, Alberto, 25

Flammini, Francesco, 3

G

Górski, Janusz, 71

Gossen, Frederik, 123

H

Hu, Shiyang, viii

I

Iantovics, László Barna, 3

M

Margaria, Tiziana, 123

Meda, Ermete, 49

Mokalled, Hassan, 49

N

Netkachov, Oleksandr, 89

Neubauer, Johannes, 123

P

Papp, József, 3

Popov, Peter, 89

Pragliola, Concetta, 49

R

Rocchetto, Marco, 25

Rushi, Julian L., 151

S

Salako, Kizito, 89

Senni, Valerio, 25

Steffen, Bernhard, 123

T

Tarani, Fabio, 177

Tokody, Dániel, 3

V

Vicario, Enrico, 177

W

Wang, Wei, 195

Wardziński, Andrzej, 71

Z

Zio, Enrico, 195

Zunino, Rodolfo, 49