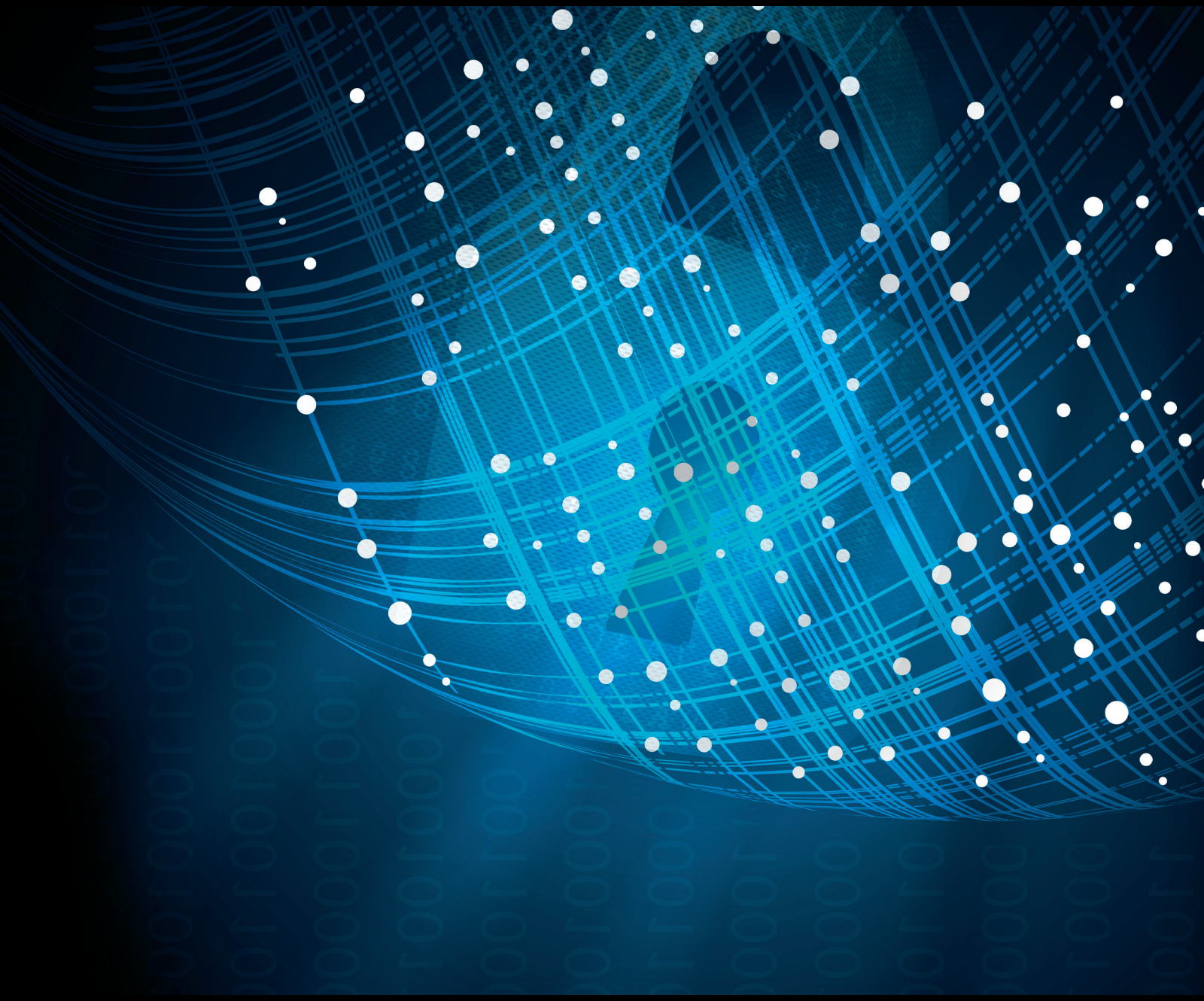
Security and Communication Networks

Safety and Security Coengineering in Embedded Systems

Lead Guest Editor: Daniel Schneider
Guest Editors: Jens Braband, Erwin Schoitsch, Sascha Uhrig, and Stefan Katzenbeisser



Safety and Security Coengineering in Embedded Systems

Security and Communication Networks

Safety and Security Coengineering in Embedded Systems

Lead Guest Editor: Daniel Schneider
Guest Editors: Jens Braband, Erwin Schoitsch, Sascha Uhrig, and Stefan Katzenbeisser

Copyright © 2019 Hindawi. All rights reserved.

This is a special issue published in "Security and Communication Networks." All articles are open access articles distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Editorial Board

Mamoun Alazab, Australia
Cristina Alcaraz, Spain
Frederik Armknecht, Germany
Benjamin Aziz, UK
Alessandro Barenghi, Italy
Pablo Garcia Bringas, Spain
Michele Bugliesi, Italy
Pino Caballero-Gil, Spain
Tom Chen, UK
Kim-Kwang Raymond Choo, USA
Stelvio Cimato, Italy
Vincenzo Conti, Italy
Luigi Coppolino, Italy
Salvatore D'Antonio, Italy
Paolo D'Arco, Italy
José María de Fuentes, Spain
Alfredo De Santis, Italy
Angel M. Del Rey, Spain
Roberto Di Pietro, France
Jesús Díaz-Verdejo, Spain
Nicola Dragoni, Denmark
Carmen Fernandez-Gago, Spain
Clemente Galdi, Italy
Dimitrios Geneiatakis, Italy
Bela Genge, Romania
Debasis Giri, India
Prosanta Gope, UK
Francesco Gringoli, Italy
Jiankun Hu, Australia
Ray Huang, Taiwan
Tao Jiang, China
Minho Jo, Republic of Korea
Bruce M. Kapron, Canada
Kiseon Kim, Republic of Korea
Sanjeev Kumar, USA
Maryline Laurent, France
Jong-Hyouk Lee, Republic of Korea
Huaizhi Li, USA
Kaitai Liang, UK
Zhe Liu, Canada
Pascal Lorenz, France
Leandros Maglaras, UK
Emanuele Maiorana, Italy
Vincente Martin, Spain
Fabio Martinelli, Italy
Barbara Masucci, Italy
Jimson Mathew, UK
David Megias, Spain
Leonardo Mostarda, Italy
Qiang Ni, UK
Petros Nicopolitidis, Greece
A. Peinado, Spain
Gerardo Pelosi, Italy
Gregorio Martinez Perez, Spain
Pedro Peris-Lopez, Spain
Kai Rannenberg, Germany
Francesco Regazzoni, Switzerland
Salvatore Sorce, Italy
Angelo Spognardi, Italy
Sana Ullah, Saudi Arabia
Guojun Wang, China
Zheng Yan, China
Qing Yang, USA
Kuo-Hui Yeh, Taiwan
Sherali Zeadally, USA
Zonghua Zhang, France

Contents

Hindawi
Security and Communication Networks
Volume 2019, Article ID 5381856, 2 pages
https://doi.org/10.1155/2019/5381856

Editorial

Safety and Security Coengineering in Embedded Systems

Daniel Schneider,[1] Jens Braband,[2] Erwin Schoitsch,[3] Sascha Uhrig,[4] and Stefan Katzenbeisser[5]

[1] *Fraunhofer IESE, Germany*
[2] *Siemens AG, Germany*
[3] *AIT Austrian Institute of Technology, Austria*
[4] *Airbus, Germany*
[5] *University of Passau, Germany*

Correspondence should be addressed to Daniel Schneider; daniel.schneider@iese.fraunhofer.de

Received 4 July 2019; Accepted 4 July 2019; Published 24 July 2019

Copyright © 2019 Daniel Schneider et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Driven by large-scale scientific, technological, and socioeconomic developments, virtually every domain of embedded (cyberphysical) systems is presently subject to the same megatrends of increasing levels of interconnection and cooperation.

In the automotive domain, future cars will be highly automated and will cooperate to optimize the overall performance (consider, e.g., traffic flow or platooning scenarios) and to prevent accidents (consider, e.g., warnings of obstacles on the road or assistance services aimed at increasing general awareness with respect to the driving behavior and intentions of other cars, or, an even more complex issue, the behavior of vulnerable road users in urban scenarios). This opens up diverse security attack vectors, and these attacks may well affect the safety of the overall system. In the railway domain, for instance, the European Train Control System (ETCS), part of the ERTMS (European Rail Transport Management System), provides high interoperability and standardized communication and control, replacing the large number of national train control systems. Any security vulnerability would be extremely critical to railway safety. Regarding the upcoming topic of air taxis and large drones, the next air traffic management systems will be highly autonomous and rely on safe, secure, and reliable communication links between air taxi vehicles and ground stations. Hence, any security issue at these links—the ground stations or the air vehicles—will cause serious safety issues across the entire avionics domain, affecting not only local air taxis or large drones, but also general aviation as well as large airliners in the vicinity of airports. In the manufacturing and process industry domain, highly automated and partially autonomous systems of all kinds are interconnected and exchange critical data. In this domain, cyberattacks may lead to safety-critical incidents with a high impact on people, the economy, and the environment. The dependency of our society on electric energy (smart grid) or other large-scale infrastructures (gas, water, and communications) leads to the same critical implications.

Thus, in the domain of safety-critical embedded systems of systems, we presently see very high potential in new cooperation-based applications and services, but we also see significant engineering challenges regarding the indispensable assurance of trustworthiness of these systems. In particular, from a safety perspective, basic assumptions like the predictability of system behavior and environment, which are the foundation of state-of-the-practice approaches and established standards, are not sufficient anymore. One reason for this is that the significant increase in communication links (connectivity) and the potential dynamic integration of insecure systems as well as the reconfiguration in adaptive open systems provide plenty of attack surfaces from a cybersecurity point of view. However, a safety-critical system that is not secure might also be not sufficiently safe, which in turn can have an impact on the placement of the product in the market.

Consequently, safety can no longer be engineered in isolation from security, and we need integrated approaches with respect to the analysis, engineering, and validation of

safe and secure cyberphysical systems of systems. Moreover, we also need specific security mechanisms that are suitable for the considered type of systems. This special issue comprises contributions with respect to each of the aforementioned dimensions; it includes contributions on the security risk assessment of connected vehicles, on security requirements engineering for a safety-critical system, on safe and secure architectures, and on security validation utilizing fault injection, as well as on a range of concrete security mechanisms/approaches in the context of cyberphysical systems of systems. The latter consist of a secure Software Defined Network (SDN)-based communication protocol, software-based memory protection, the application of physically unclonable functions to defeat manipulation attacks in the context of tiny IoT devices, and other measures.

We conclude our editorial by briefly summarizing these different contributions to this special issue. First, there are two articles focusing on risk assessment and requirements engineering, respectively.

In their article "A Comparative Study of JASO TP15002-Based Security Risk Assessment Methods for Connected Vehicle System Design", the authors introduce the idea of asset containers and propose extending CRSS (Common Vulnerability Scoring System (CVSS) based Risk Scoring System) to a novel Risk Scoring System (RSS)s, RSS-CVSSv3, by appropriately replacing the CVSSv2 vulnerability scoring system on which CRSS is based with CVSSv3. To address the above questions, they performed a comparative study on CRSS, RSMA (Risk Scoring Methodology for Automotive systems), and RSS-CVSSv3 for multiple use cases such as a CGW (Central Gateway) and a drone in order to examine the efficiency and usefulness of the presented methods. For this comparative purpose, they devised an approach for the refinement of RSMA to the obstacles in comparing CRSS with RSMA.

In the article "Security Requirements Engineering in Safety-Critical Railway Signalling Networks", the authors report on their experience in developing security architecture for railway signalling systems, starting from the bare safety-critical system that requires protection. They used a threat-based approach to determine security risk acceptance criteria and derive security requirements and developed a security architecture based on the security requirements. The architecture is based on a hardware platform that provides the resources required for safety as well as security applications and is able to run these mixed-criticality applications (safety-critical applications and other applications run on the same device). To achieve this, the MILS approach is applied and discussed in detail in relation to the security requirements.

With respect to security validation of (safety-critical) systems, there is an article entitled "Multidevice False Data Injection Attack Models of ADS-B Multilateration Systems". The authors assume that attackers equipped with multiple devices can manipulate the ADS-B (Automatic Dependent Surveillance Broadcast) messages in distributed receivers without any mutual interference and can thus efficiently construct attack vectors to change the results of multilateration. The feasibility of a multidevice false data injection attack is demonstrated experimentally and countermeasures for such attacks are discussed.

Last but not least, three contributions deal with concrete security mechanisms/approaches for the context of cyberphysical systems of systems.

A secure SDN-based protocol is presented in "SSPSoC: A Secure SDN-Based Protocol over MPSoC". Following the SDN concept, the authors propose a new protocol in order to secure the communication and efficiently manage the routing within the CoC (Cloud of Chips). The SSPSoC includes a private key derivation phase, a Group Key Agreement (GKA) phase, and a data exchange phase to ensure that basic security primitives are preserved and provide secure communication. Furthermore, a network of 1-30 nodes is used to validate the proposed protocol and measure the network performance and memory consumption of the proposed protocol.

Next, there is an article focusing on memory protection, "SoftME: A Software-based Memory Protection Approach for TEE System to Resist Physical Attacks". The approach utilizes the on-chip memory space to provide a trusted execution environment for sensitive applications. It uses data encryption to protect the confidentiality of data on the off-chip memory and to provide integrity protection for the data. In addition, task scheduling in the encryption process is implemented. The prototype system of the approach was implemented on a development board supporting TrustZone and the overhead of the approach was tested. The experimental results show that the approach improves the security of the system and that there is no significant increase in system overhead.

In their article "Single-Round Pattern Matching Key Generation Using Physically Unclonable Function", the authors discuss that this not only enables defeating manipulation attacks but also makes it possible to prove security theoretically. In addition to its simple construction, the utilized scheme can use a weak PUF like the SRAM-PUF as a building block if the system is properly implemented, so that the PUF is directly inaccessible from the outside. It is therefore suitable for tiny devices in IoT systems. The article discusses the scheme's security and demonstrates its feasibility by means of simulations and experiments.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

We thank the authors for their contributions and the reviewers for their valuable comments, which made this special issue possible.

Daniel Schneider
Jens Braband
Erwin Schoitsch
Sascha Uhrig
Stefan Katzenbeisser

Hindawi
Security and Communication Networks
Volume 2019, Article ID 8348925, 14 pages
https://doi.org/10.1155/2019/8348925

Research Article

Security Requirements Engineering in Safety-Critical Railway Signalling Networks

Markus Heinrich,[1] Tsvetoslava Vateva-Gurova,[1] Tolga Arul,[1] Stefan Katzenbeisser,[1] Neeraj Suri,[1] Henk Birkholz,[2] Andreas Fuchs,[2] Christoph Krauß,[2] Maria Zhdanova,[2] Don Kuzhiyelil,[3] Sergey Tverdyshev,[3] and Christian Schlehuber[4]

[1]*Department of Computer Science, TU Darmstadt, Germany*
[2]*Fraunhofer Institute for Secure Information Technology (SIT), Germany*
[3]*SYSGO AG, Germany*
[4]*DB Netz AG, Germany*

Correspondence should be addressed to Markus Heinrich; heinrich@seceng.informatik.tu-darmstadt.de

Received 21 December 2018; Revised 20 March 2019; Accepted 16 May 2019; Published 14 July 2019

Academic Editor: David Megias

Copyright © 2019 Markus Heinrich et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Securing a safety-critical system is a challenging task, because safety requirements have to be considered alongside security controls. We report on our experience to develop a security architecture for railway signalling systems starting from the bare safety-critical system that requires protection. We use a threat-based approach to determine security risk acceptance criteria and derive security requirements. We discuss the executed process and make suggestions for improvements. Based on the security requirements, we develop a security architecture. The architecture is based on a hardware platform that provides the resources required for safety as well as security applications and is able to run these applications of mixed-criticality (safety-critical applications and other applications run on the same device). To achieve this, we apply the MILS approach, a separation-based high-assurance security architecture to simplify the safety case and security case of our approach. We describe the assurance requirements of the separation kernel subcomponent, which represents the key component of the MILS architecture. We further discuss the security measures of our architecture that are included to protect the safety-critical application from cyberattacks.

1. Introduction

The integration of commercial off-the-shelf (COTS) hardware and software into industrial control systems such as railway command and control systems (CCSs) is in progress. However, introducing COTS components into a previously proprietary safety system leads to novel security threats. The interplay between safety and security is an active research area, where many questions are yet to be answered. An extensive survey of approaches to combine safety and security in industrial control systems has been performed by Kriaa et al. [1]. Our study of the safety-security interplay is motivated by the lack of a security architecture for railway signalling.

Current train control is centralized in a signal box, also called interlocking system (ILS), that controls a defined area of the tracks comprising of multiple track switches (points) and signals. An example with a single point and signal is shown in Figure 1. If a train needs to move on the tracks, the ILS sets the points according to the desired route. If the movement of the train is considered safe, the ILS sets the signal for the route to clear. The aspect of the signal is observed by the train driver, who is allowed to safely proceed on the journey. A route is considered safe for a train, if it is not occupied by or reserved for another train, thus precluding collisions.

Points, signals, and other controllable objects are summarized under the term field elements. Earlier ILS generations used analogue signal transmission to set their field elements. Modern ILSs utilize IP networks to transmit their commands digitally to an object controller (OC) that in turn steers the

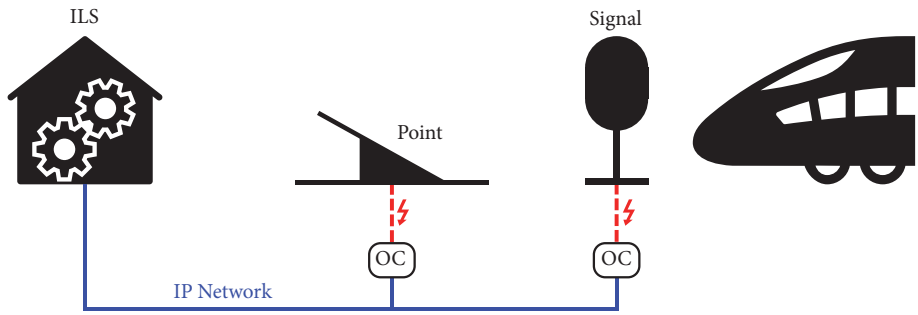


Figure 1: A railway command and control system.

field element by starting and stopping the point machine or turning on and off the signal's light bulbs, respectively. This allows for decoupling energy supply and command transmission and thus for larger distances between ILS and field element. Since railway signalling networks are classified as critical infrastructure (CI), the impact of potential security incidents on the railway system can be huge. This calls for the need of a security concept to ensure the robustness of railway signalling networks against cyberattacks. Furthermore, the railway system as a CI must meet national safety regulations. To address these issues, we execute a railway domain-specific requirements engineering process that has been proposed for German railways [2] but can be used as a template for international railway operation. The output of this process provides the foundation of a security architecture, which we propose for railway CCS.

Our contribution consists of the following. We report on our experience with the requirements engineering process of DIN VDE V 0831-104, and make suggestions for improvements. Then, we investigate the effect of these requirements on our case study, the safety-critical railway CCS. We show and discuss the derived security requirements for the case study. Subsequently, we use the identified threats and requirements to propose a security architecture for mixed-criticality systems such as the railway CCS running safety-critical applications along nonsafety-critical security applications. A mixed-criticality system must be carefully designed in order to maintain functionalities such as dependability and responsiveness under constrained resources and in the presence of attackers [3]. We propose the Secure Object Controller (SecOC), a security architecture based on a hardware platform that includes a hardware trust anchor. On top of the hardware, a separation kernel (SK) provides a software framework that allows running applications of mixed-criticality on our platform. On the software platform, safety and security applications coexist. The security applications protect the safety-critical application from cyberattacks. Complementary, the SK ensures that the security applications cannot exhaust the resources required by the safety application to fulfill its safety-critical task. Additionally, to enhance the security of the system, we apply security measures. An authenticated boot process uses the hardware trust anchor to ensure that only authorized software instances are executed on the hardware platform. A health monitor observes the system state during runtime and can report conspicuous state changes. A secure update mechanism allows for altering the system software, firmware, and configuration from authorized sources only. We further discuss our approach to handle assurance of the mixed-criticality system towards multiple certification authorities. Also, we show how the proposed architecture fulfills the requirements we derived.

The paper is organized as follows. Section 2 presents the system model. Section 3 details the conducted threat analysis. The elicited security requirements are presented in Section 4. Section 5 discusses the shortcomings of the conducted process as well as the suggested enhancements. Section 6 describes and discusses the proposed security architecture. Section 7 justifies that our security architecture fulfills the security requirements. Section 8 concludes the paper.

2. System Model

Our system model is a general railway signalling architecture composed of the interlocking layer and the field element layer, as depicted in Figure 2.

The interlocking layer encompasses the ILS and a maintenance and data management system (MDM). The MDM is responsible for updating the software of the components in the interlocking and field element layer as well as for time synchronization, and diagnostic data collection due to legal requirements. The heart of the signalling network is the ILS. It is responsible for issuing commands to the field elements of the lower architectural layer to execute the orders of the traffic supervisor, who is not considered in our system model. The ILS guarantees safe train operation by discarding unsafe orders, e.g., orders that would lead to the collision or derailment of trains.

On the lower layer, point machines are driven to switch tracks and clearances are communicated to train drivers via light signals. The commands to the OCs are sent through the network using specialized railway protocols. The OCs, located in junction boxes close to the tracks, switch the power of their corresponding field element (points, signals). There is a one-to-one relation between field element and OC. Thus, OCs are spatially distributed with limited physical protection by junction boxes and are therefore accessible by an attacker. In contrast, physical access to the components of the interlocking layer is more difficult to obtain, as

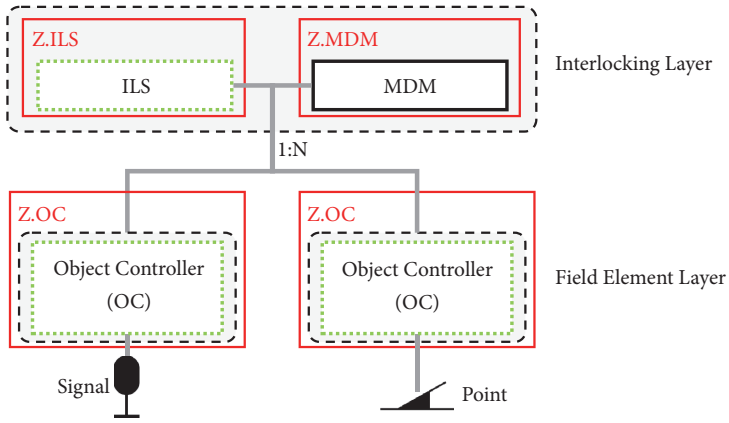


Figure 2: System model showing `Z.ILS`, `Z.MDM`, and two instances of `Z.OC`. Dashed frames indicate housing. Dotted frames indicate existing safety-critical functionality.

these components are subject to physical access control in a building.

The layers are connected via a network that is considered untrusted from a security perspective, because the railway operator has neither full control over its physical layout nor can guarantee its impenetrable protection. As a result, packets might be dropped, changed, rerouted, or read by an unknown third party.

The depicted building blocks are located in housings (indicated by the dashed line in Figure 2). This fact is important for the subsequent security analysis, because of the influence the housing can have on the attacker's capabilities to access the components. It protects the internal network of the interlocking layer and the enclosed objects such that certain malicious actions, e.g., physical penetration, become infeasible.

According to DIN VDE V 0831-104 [2], we split the system model into logical zones featuring components that are assumed to have equal security requirements. We identify three zones, named `Z.ILS`, `Z.MDM`, and `Z.OC`, shown in Figure 2. Our definition of zones includes the list of objects in the zone, the logical and physical borders, the data flows between the zones, the interfaces (Ethernet), and the physical connections of each zone. In our system definition, zones `Z.ILS` and `Z.MDM` reside in the same building and are instantiated only once per railway station. Many zones of type `Z.OC` exist in a single railway station and have weaker housing compared to `Z.ILS` and `Z.MDM`. The number of `Z.OC` is determined by the number of field elements controlled by a railway station and can reach up to 250 entities.

We use the system model consisting of the three zones and their interconnections as input for the next process step where we analyse the threats to the system.

3. Risk Analysis

To derive security requirements, we follow the German prestandard DIN VDE V 0831-104 for security in railway signalling networks [2]. It is designed as a guideline for applying the IEC 62443 framework for industrial control system security to railway signalling, while obeying to relevant railway safety standards such as EN 50128, EN 50129, and EN 50159.

In this section, we detail the conducted risk analysis, considering each of the zones of the system model (see Section 2) along with its components, their functions, the relevant data, and the communication between the components.

From a safety perspective, the risk is indicated as the product of the probability of an event to occur and the loss associated with the event. This approach has proven to be impractical for security analyses [4]. Attacks on security do not follow a probabilistic model of occurrence, because they are intentional. Moreover, even if security attacks could be characterized by a stochastic process, probabilities could not be easily determined due to the lack of a sufficient amount of samples and a constantly changing attacker landscape. Therefore, we conduct an explicit risk analysis, following one of the approaches proposed by DIN VDE V 0831-104, by using the presented system model.

3.1. Attacker Model. In order to describe capabilities of an attacker, DIN VDE V 0831-104 uses a set of qualitative dimensions: resource, knowledge, and motivation. The *resource* dimension reflects the financial and workforce capacity of the attacker to prepare and launch an attack. The *knowledge* dimension describes what she knows about the system before attacking it and can use to create opportunities for a successful offensive. These dimensions are characterized by numerical values from low (2) over medium (3) to high (4), where attackers with basic capabilities (1) are not considered [2]. Thus, *generic* knowledge (K = 2) comprises of publicly available information such as protocol specifications and COTS hardware. An attacker with *extended* knowledge (K = 4) has access to some closed information usually only available to a small circle of experts (insiders) working with the system. *Low* financial resources (R = 2) comprise a few thousand Euro while *extended* financial capacity (R = 4) provides resources in the magnitude of state actors.

Table 1: Examples of identified threats with assigned values for the attacker. FR: foundational requirement, R: resources, K: knowledge, PSL: preliminary security level, LOC: location, TRA: traceability, EXT: extent of damage, SL: security level.

Threat	FR	R	K	PSL	LOC	TRA	EXT	SL
`T.SI.Attacker.Malware`	SI	3	4	4	0	0	0	4
`T.RA.Attacker.DoS`	RA	2	2	2	0	0	1	1
`T.DC.Attacker.TrafficAnalysis`	DC	2	2	2	1	0	1	1

In order to describe the attacker's motivation DIN VDE V 0831-104 introduces railway-specific mitigation factors related to the risk for the attacker to be discovered. The higher this risk for a particular attack, the lower is the motivation to carry it out. This way, existing security controls, can be taken into account.

The standard considers the following mitigation factors: location (LOC), traceability (TRA), and extent of attack (EXT) [2]. LOC determines whether an attack can be executed remotely (LOC = 0) or only given a physical access to the system (LOC = 1). Remote attacks are considered to be more dangerous, as the chance for the attacker to remain undiscovered is higher. Similarly, if an attack can be traced to the attacker (TRA = 1), it is less dangerous than an untraceable attack (TRA = 0). EXT denotes the potential damage of an attack, which can be either *critical* (EXT = 1) or *serious* in case fatalities might be involved (EXT = 0).

This attacker model is applied during the risk assessment to evaluate each potential threat and determine the security level required to protect the system against it (see Section 3.2). Though this approach does not explicitly employ traditional attacker categories including, e.g., basic user, cybercriminals, terrorists, insiders, or nation state [5], it uses the same idea to describe attacker's capabilities.

In the specific context of this paper involving safety aspects, the only relevant goal of the attacker is to cause collisions or derailment of trains. This goal can also be achieved by simply blocking the tracks, exchanging the colors of a light signal or detaching the light signal from the OC and applying current, so that the signal shows clear when it should show stop. For example, political activists from Greenpeace are reported to cause disruptions by sabotaging the railway to prevent the transportation of nuclear waste (http://www.greenpeace.org/international/en/news/Blogs/nuclear-reaction/greenpeace-block-nuclear-waste-transport/blog/35204/). This kind of simple physical attacks targeted against individual field elements have always been possible and cannot be fully prevented by IT security mechanisms.

The digitalization in turn introduces much broader adversarial opportunities due to connectivity and usage of regular IT components. Thus, our goal is to protect the railway signalling network against *one-affects-all* attacks that can be applied to a multitude of field elements at the same time and can paralyze the complete infrastructure. Additionally, by the use of COTS products, attacks can become remotely executable in case the attacker finds a way to access the network shown in Figures 1 and 2. In contrast to such scenarios, physical attacks are much more restricted. As the hardware belonging to `Z.ILS` and `Z.MDM` is set up in a building with physical access control, it is reasonable to assume that an attacker cannot gain physical access to this hardware to modify or even replace the components. The OCs in `Z.OC` are installed in a junction box and therefore physically accessible for the attacker to tamper with the hardware or replace it entirely. However, we assume that physically tampering with an OC is equivalent to local attacks we described before (blocking the tracks; changing colors of light signal). They do not scale to a one-affects-all attack, because each field element has to be attacked individually. Also it is considered virtually impossible to protect against such physical attacks in a large-scale infrastructure like nationwide railway signalling [6, 7]. For this reason, only protection against high-impact one-affects-all attacks is the main scope of this paper.

3.2. Threat Analysis and Preliminary Security Level. We perform a threat-based risk analysis as defined in DIN VDE V 0831-104 by systematically listing cyberattacks that threaten the components presented in our system model. This approach allows us to consider typical cyberattacks as well as infrastructure-specific and railway-specific attacks. As a result of the threat analysis, 67 threats are defined.

Some examples are given in Table 1. The table also shows the underlying attacker model of DIN VDE V 0831-104 as explained in the previous section.

Each threat was given an identifying name and a detailed description, where the description provides details about attack implementation and potential impact. For example, `T.SI.Attacker.Malware` describes an attacker who introduces malware to undermine the integrity of the railway CCS. A typical denial-of-service (DoS) attack is covered by the threat named `T.RA.Attacker.DoS`. A threat more typical for railway CCS describes an attacker that records and analyses the traffic of a signalling network in order to prepare a more sophisticated attack (`T.DC.Attacker.TrafficAnalysis`).

The threats are assigned to at least one of the foundational requirements (FRs) of IEC 62443 being identification and authentication control (IAC), use control (UC), system integrity (SI), data confidentiality (DC), restricted data flow (RDF), timely response to events (TRE), and resource availability (RA).

In addition, each threat was assigned to the zones to which it is applicable. Out of all the 67 threats, 51 are applicable to `Z.OC`, 51 are applicable to `Z.ILS` and 47 are applicable to `Z.MDM`. While 38 threats are relevant for all the three zones (i.e., executable on different components), seven are applicable to exactly two zones, and 22 are specific to a single zone (i.e., exploiting the characteristics of the respective zone).

Table 2: SL vectors of the zones as the result of the analysis.

Zone	IAC	UC	SI	DC	RDF	TRE	RA
`Z.OC`	3	2	*4*	1	1	3	1
`Z.ILS`	3	2	*4*	1	1	3	1
`Z.MDM`	3	2	*4*	1	1	3	1

To estimate the risk imposed by a given threat, we consider the attacker capabilities for this threat in accordance with the attacker model introduced in Section 3.1. The attacker's resource and knowledge capabilities are combined to form a preliminary security level (PSL) related to the given threat. The PSL is later used to calculate the final security level (SL) for the respective threat. A SL—ranging from 1 (low) to 4 (high)—describes the level of protection a system provides against an attacker.

During the execution of the risk assessment process each threat is assigned values for the attacker capabilities (R, K) and values for the mitigation factors (LOC, TRA, EXT) to calculate and respectively adjust the PSL relevant for the respective threat. Examples are shown in Table 1.

3.3. Calculation of Security Levels. As defined by DIN VDE V 0831-104 [2], the final SL, using the PSL and the mitigation factors, is calculated by

$$\mathrm{SL} = \mathrm{PSL} - \max\{\mathrm{LOC}, \mathrm{TRA}, \mathrm{EXT}\}. \quad (1)$$

According to the equation, the value of the PSL is reduced by one, if any of the mitigation factors equals one (corresponding to a logical "or"). This means that a threat that can only be executed locally (LOC) is traceable (TRA) or has only a critical (*critical* in this case, is the less dangerous/severe outcome) extent (EXT) which will reduce the PSL by one level to form the SL.

This calculation is done for each of the 67 identified threats in the seven FRs. To calculate the SL value for the three zones, each is assigned a vector of seven values corresponding to the respective FRs, as shown in Table 2. The seven entries are determined by the maximum SL value over the threats assigned to the respective zone and FR. For each zone, the FR with the greatest SL determines the security level of the zone (set italic in Table 2). For simplicity, we write SL x, when we refer to an SL vector with maximum entry x. In this way, the SL vector of `Z.OC` $(3, 2, 4, 1, 1, 3, 1)$ yields SL4 for the zone. Respectively do the vectors of `Z.ILS` and `Z.MDM` yield an SL of 4.

We identify three decisive threats that determine the security levels. We use them to exemplify how the attacker capabilities and mitigation factors lead to the SL. Two of the decisive threats are responsible for the value of 4 in all three zones. The first decisive threat is the remote execution of malware on the systems in our reference architecture (`T.SI.Attacker.Malware`, see Table 1). We assigned an attacker with moderate resources (R = 3) and extended knowledge (K = 4) to it, resulting in PSL = 4 for the threat. The assessment of the mitigation factors resulted in the following values: the threat description implies that it is remotely executable (LOC = 0). A skilled attacker is assumed to be able to hide the traces, such that we consider the threat as not traceable (TRA = 0). Also, an attacker with deep knowledge (K = 4) is capable of performing a carefully targeted attack with potentially serious extent (EXT = 0). Thus, none of the mitigation factors apply to reduce the final SL. Subsequently, plugging the mitigation factors and the PSL into (1) yields SL4.

The second decisive threat describes the manipulation of patches such that legitimate processes execute malicious code chosen by the attacker when rolled out to devices through update mechanisms. Analogous considerations for attacker capabilities and mitigation factors as in the previously discussed case apply to this threat leading to SL 4.

Only for zone `Z.MDM` there is a third decisive threat that results in SL 4. It covers the manipulation of data on the MDM where the firmware of the ILS and OCs is stored and distributed from. This threat poses a high risk, because the firmware can be manipulated remotely at a central point from which it is distributed to unsuspecting network entities. Without further checks, the ILS and OCs of an entire station's signalling network can be compromised. Again, we consider an attacker with R3 and K4 to perform this attack and could not identify an applicable mitigation factor (LOC = 0, TRA = 0, and EXT = 0). Hence, the analysis of this threat also leads to SL 4 for zone `Z.MDM`.

4. Security Requirements Elicitation

After having derived the SL of each zone, the specific system requirements can be retrieved from IEC 62443-3-3. The standard contains a list of 100 system requirements that are applicable to each zone depending on the identified SL. We evaluated the SL for each FR to select the security requirements from the IEC 62443-3-3 standard. As a result of our risk analysis, we found 69 system requirements that are relevant for our system model.

The requirements of IEC 62443 are on a generic level, as it is a standard for industrial automation and control system (IACS). In order to reduce the complexity of handling 69 relevant requirements and to conserve the railway-specific knowledge gathered while specifying the threats, we choose to explore an additional approach to derive security requirements. For this approach, we elicit requirements with a methodology suggested by Myagmar et al. [8]. The authors propose to take the outcome of the conducted threat analysis as a basis, and transform each identified threat to a requirement. Deriving requirements besides IEC 62443 opens the possibility of responding to railway-specific factors that cannot be reflected in a IACS standard like IEC 62443.

Examples for such railway-specific requirements are R1, R11, and R13 that are shown and discussed later in this section.

In order to derive requirements, we investigate corresponding threats and formulate a statement indicating which behaviour the system must or must not exhibit in order to prevent the respective threat [8]. We mark the threat to be covered by the respective requirement. Alternatively, we declare that the required behaviour was already contained in a previously found statement, so we mark the threat to be covered by the respective requirement. We repeat the addition of requirements until every threat has at least one corresponding requirement. Subsequently, we derive the following security requirements that are explained in more detail afterwards:

- R1 The system shall detect unauthorized physical access to its subsystems and/or prevent relevant exploitations of physical access.
- R2 The system shall not allow the compromise of a communication key.
- R3 The system shall not disclose classified or confidential data (such as access credentials) to any illegitimate user.
- R4 The system shall exclude compromised endpoints from communication.
- R5 The system shall not use insecure transfer methods.
- R6 The system shall not allow any unauthorized user to access an endpoint (e.g., MDM, ILS, and OC).
- R7 The system shall not allow unauthorized and unauthenticated communication between endpoints.
- R8 The system shall not violate the runtime behaviour requirements.
- R9 The system shall allow for the updating of security mechanisms, credentials, and configurations in order to patch known vulnerabilities.
- R10 The system shall not allow the execution of unauthorized software instances.
- R11 The system shall maintain the transmission system requirements defined in EN 50159.
- R12 The system shall provide mechanisms to detect an undesirable system state change and anomalies.
- R13 The system shall impede that an unauthorized user can force it into one of the fall-back levels defined by the railway safety process.
- R14 The system shall maintain the integrity of software, firmware, configuration, and hardware.

In the following, we discuss those requirements that are specific to railways as they could violate the safety constraints posed by the domain standards.

The physical access detection required by R1 is especially relevant for the railway domain, because OCs are spatially distributed over large areas next to railway tracks and cannot be as well protected as, for example, within factory premises. Therefore, unauthorized physical access to the junction box of the OC has at least to be detected such that further actions can be triggered by, e.g., a security operations center to avoid or mitigate consequences.

In order to keep a railway station operational, R4 requires that compromised endpoints (OCs) can be excluded from the network, such that benign OCs are not affected. An endpoint is considered compromised, if an attacker can remotely control the safety functionality, because the OC accepts the attacker's commands or the attacker controls the safety-critical software on the endpoint. The disclosure of cryptographic keys belonging to an endpoint renders it compromised as well. Physical access to the OC's hardware constitutes a compromise that can be detected if the junction box is opened or the OC is removed from the rack. Physical access to the steered field element is not a compromise as this attack is already possible in current railway infrastructures without digital components. However, physical attacks of this kind do not scale to multiple OCs, if no shared secrets can be gained by the attacker.

Requirement R5 excludes transfer methods such as network protocols that involve cryptographic functions which usage is discourage by institutions like NIST or national agencies for information security. The requirement enforces the usage of communication protocols that do not employ cryptography that is considered broken.

Due to safety reasons, railway signalling networks require a failure disclosure time, which is addressed by R8. Any security measure that influences the network traffic (e.g., message encryption) must not exhaust the network resources such that the network latency exceeds a threshold of 50 ms as specified by railway safety standards and railway operator specifications.

A common requirement from both security and safety perspectives is the robustness of a transmission system against repetition, deletion, insertion, resequencing, corruption, delay, and masquerade of messages. This is specified by EN 50159, a European standard for safety-related communication in transmission systems required to receive admission to operate a railway system. R11 ensures that these requirements are fulfilled and also considered in the design of a security architecture to avoid fulfilling a requirement twice. Some security functionalities might already be available in the system due to fulfilling EN 50159 or can be established by adding only minimal features.

During the design of the security architecture, it must be considered that railway signalling has processes that take effect in case of technical failure in order to maintain operation of the railway system. These fall-back processes involve human interaction and do not provide the full extent of safety and capacity compared to a fully functional, automated interlocking in terms of failure rate. This risk is covered by R13, which requires the security architecture to be designed in such a way that it does not allow an attacker to force the interlocking system into a fall-back state. The attack surface should not be increased by the security architecture compared to attacks with the same effect already possible today, e.g., physically destroying a cable. A security architecture that can force the safety system into a fall-back state enables the attacker to remotely implement a

large-scale DoS attack causing major disruption in the railway transportation system.

The derived requirements will be used to define a security architecture for our system model. The security concept will be outlined in Section 6.

5. Discussion of the Risk Analysis Process

While conducting the risk analysis process according to DIN VDE V 0831-104, we found some shortcomings of the standards that should be addressed. We also make suggestions how the process could be improved. Moreover, we analyse how the suggestions would influence the resulting requirements.

5.1. Disjunction of Mitigation Factors. Equation (1) allows reducing the PSL to obtain the SL if any of the mitigation factors is applicable. This means that a lower security level is found if one of the following conditions for an attack (as the execution of a threat) is met: The attack either can only be performed locally, is traceable to the attacker, or is not expected to cause fatalities. This contradicts intuitive human risk acceptance. A threat that most certainly causes fatalities but is only locally executable leads to at most SL 3 for the associated zone. This is because the PSL will be reduced by 1 through the mitigation factor LOC = 1. An attack with the same type of attacker (same level of knowledge and resources), that is also only locally executable (LOC = 1) and additionally traceable (TRA = 1) but will not cause fatalities (EXT = 1), will lead to the same security level as the previously constructed threat and will require the same security protection as a result. Whether human lives could be lost due to this attack is not valued in this scenario, because the application of one mitigation factor is sufficient to reduce the SL.

To eliminate this contradiction, we suggest reducing the SL only if all three railway-specific mitigation factors apply. This is reflected by (2), where, instead of the maximum, the minimum over the mitigation factors is subtracted from the PSL:

$$\mathrm{SL} = \mathrm{PSL} - \min\{\mathrm{LOC}, \mathrm{TRA}, \mathrm{EXT}\}. \tag{2}$$

We recalculate our threat analysis described in Section 3.3 according to (2) and find now 19 instead of 2 threats with SL 4. Overall, the SL of 54 of the 67 threats is increased by one, leaving 13 unchanged. Since SL 4 threats existed in our previous analysis, the final SL does not change, but has now greater support with 19 threats instead of 3.

5.2. Reduction of Security Level. The standard DIN VDE V 0831-104 does not state whether it is allowed to apply the mitigation factors to a PSL of 2 or 1. Such a reduction would result in zones which have SL 1 and SL 0, respectively. Following DIN VDE V 0831-104, a zone with SL 1 only requires basic protection against unintentional or incidental attacks, which are already covered by safety measures. SL 0 does not require any security protection at all, although an attack might have been found in the previous analysis.

We suggest that the DIN VDE V 0831-104 should exclude mitigating PSL 2 to SL 1 by definition. In our analysis, this would affect 11 threats obtaining SL 2 instead of SL 1. Still, the SL of the entire zone is dependent on the strongest attack (i.e., the threat with highest SL). Therefore, it is unlikely that a threat with SL 1 is decisive for the system.

5.3. Transforming Threats. During the risk analysis of DIN VDE V 0831-104, very specific threats are collected that are based on detailed knowledge of the reference architecture. In the subsequent process of the prestandard, this railway domain-specific information is lost, as it cannot be reflected by the final requirements taken from IEC 62443, which are generic for IACS. We overcome this by deriving requirements directly from the domain-specific threats as described in Section 4. By employing this methodology, we are able to reflect the specifics of railway operation in our security requirements and maintain the knowledge gathered about potential attacks during the threat analysis. Examples of railway-specific requirements are given and explained in Section 4 based on requirements R1, R8, R11, and R13.

5.4. Assigning Maximum Entry in SL Vector. Contrary to our previously described approach, DIN VDE V 0831-104 suggests assigning the maximum entry of the SL vector to all FRs it defines as being safety relevant. The safety-relevant FRs are IAC, UC, SI, and TRE [2]. With this method our SL vector for `Z.OC` would be adjusted from $(3, 2, 4, 1, 1, 3, 1)$ to $(4, 4, 4, 1, 1, 4, 1)$ (compare Table 2). This transformation aims to reduce the complexity in SLs, because only one entry appears (besides the values defaulting to "1"). However, we believe that this adjustment of the level is not justified by the achieved simplification, as it increases the number of requirements derived from it. Applied to our analysis, the transformation of the SL vector results in 15 additional requirements compared to the 69 we identified with the SL vectors explained in Section 3.3. For instance, UC = 2 is increased to UC = 4, which results in 12 additional requirements, stemming from this FR. The additional requirements entail the implementation of further measures and additional time and financial effort that could otherwise be avoided.

In hindsight, the comprehensibility of the process' result is lost, because it is unclear whether an entry in the vector stems from an actual threat demanding it or from adjusting the values to the greatest entry. Therefore, we chose not to follow the suggestion and kept the vectors as presented in Section 3.3.

6. Software and Hardware Architectural Considerations for Safety and Security

We use the security requirements derived in Section 4 to define a hardware-based security platform for railway CCS.

In this article, we analyse the coexistence of safety and security applications that not only have different assurance levels but should be certified by different authorities competent in each domain. In our case from safety and information security authorities, most countries operate a

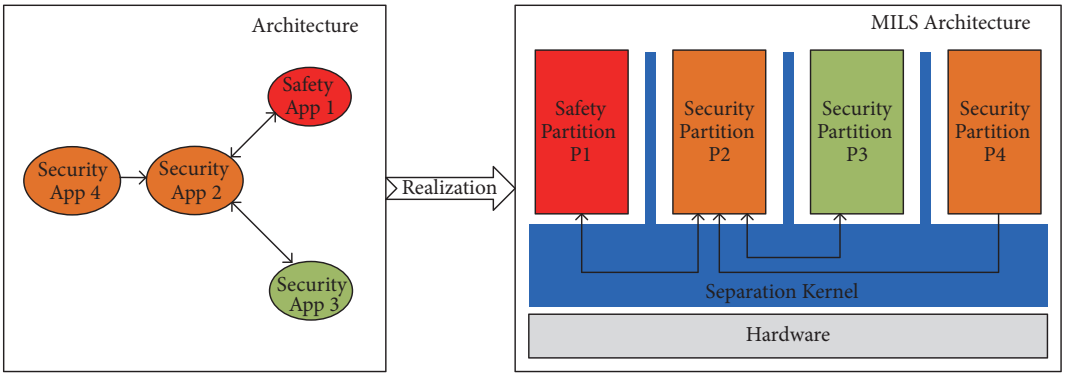


Figure 3: Proposed architecture and its realization using a Separation Kernel.

national safety authority whose admission is required to be allowed to operate a railway system (safety case). Most of the safety authorities and railway operators recognize the increasing need for security functionality and respective security certification due to the utilization of COTS products in the domain. As most safety authorities lack the experience for security certification, this task is fulfilled by respective national authorities for information security.

6.1. The Object Controller as a Mixed-Criticality System. With regard to the coexistence of safety-critical and security applications on the same platform, the railway safety standard EN 50128 requires that all software should be certified to the highest safety integrity level (SIL) unless evidence of independence (freedom of interference) can be provided [9, Section 9.4.9]. However, for multiple reasons, we do not want to treat the newly developed security applications as belonging to SIL4 like the OC's safety application. Obtaining the highest safety certification is hardly feasible for security applications since they are developed with a dynamic lifecycle, considering the ever changing attack potential of the environment that evolves much faster than any safety certification cycle. Certifying the safety-critical subsystem without the need to evaluate the security applications allows us to protect the safety system with complex security measures that can react to the changing vulnerability landscape in an adequate time frame without recertification.

Thus, from the safety certification point of view, we are building a mixed-criticality system that combines applications with different safety assurance levels on the same (hardware) platform [10]. In our case, the OC safety application with SIL4 requirements and security applications with no safety assurance requirements are combined on the same platform.

6.1.1. MILS Architecture as a Solution for the Sharing Challenge. We follow the Multiple Independent Levels of Security (MILS) approach [11–14] in order to simplify the certification described in the previous section. MILS is a high-assurance safety/security architecture based on the concepts of separation and controlled information flow. The cornerstone of the architecture is a separation mechanism that encapsulates applications of different criticality in different partitions. These partitions reduce mutual dependencies to communications over channels explicitly defined by policies. The separation mechanism has to be nonbypassable (cannot be circumvented), evaluable (small and simple enough that proof of correctness is practical and affordable), always invoked, and tamper-proof (cannot be modified without authorization) [14]. The safety and security assurances for this separation mechanism are described in Section 6.1.2

The MILS platform is implemented as a software-hardware system, consisting of a software separation kernel (SK), the hardware central processing unit (CPU), and memory management unit (MMU), as well as other critical hardware devices (e.g., I/O MMU) and their software (see Figure 3).

The central part of a MILS platform is its SK, which provides separated partitions, manages hardware, and enforces security policies for information flow, access control, and resource availability. A SK is a special kind of operating system often based on the microkernel approach [15] to achieve modularized design and minimize the trusted computing base. These properties allow for a formal verification of the kernel [16]. MILS is based on system decomposition of hardware and software into meaningful and—from a security point of view—atomic components. Thus, safety and security by design is achieved by partitioning the system on the architecture level, using the SK to enforce these architectural decisions.

The MILS terminology defines *separation* as the condition where two applications are allocated in different partitions and there is an explicit communication channel between these partitions. Partitions are used to create security domains. Communication between partitions represents the explicit crossing of domain boundaries and is realized using communication objects under the control of the SK. The information flow policy used between partitions using the communication objects is statically defined during the integration phase by specifying the allocation of communication objects to the partitions and the permissions. During runtime, the SK will enforce the allocation and access permissions as per the static configuration.

For realizing our use-case of integrating the safety and security applications on the same platform, we map

these applications to different partitions. This architectural refinement is shown in Figure 3. Depicted on the left-hand side is a high level architecture, showing the high-critical safety application in red and security applications from different domains in green and orange. The arrows between the applications indicate the communication channels and their directions. The architectural refinement using the SK is shown on the right-hand side, where the safety and security applications are assigned to safety and security partitions and the communication channels are realized using the communication objects that are under control of the SK.

With the help of the underlying SK, we make sure that the safety partition is both spatially and temporally separated from the security partitions, in order to guarantee code and data integrity as well as real-time. The hardware platform used for deploying the proposed architecture shall be fast enough to reserve CPU time required for the safety application to meet its deadlines and at the same time have remaining CPU time available to the security applications to perform their functions.

With such a well-separated MILS architecture, we keep the existing safety certification of the safety application once it is integrated together with the security applications in our new system. We pave the way to retain the safety certification when the security applications are updated in-the-field to respond to the changes in the security landscape, such as the discovery of a weakness in a cryptographic algorithm or a vulnerability in the deployed software.

6.1.2. Safety and Security Assurance for the Separation Kernel Subcomponent. As described in Section 6.1.1, the main function of SK subcomponent in the MILS architecture is to provide strong separation and well-defined information flow channels between partitions that host applications of different criticality levels. The SK can be considered as a single point of failure in the MILS architecture. Thus, the SK shall provide assurance for the highest safety and security levels required for the system.

This is described in the EN 50128 standard that states that the subcomponent that provides separation between components shall be certified at the same level of the highest assurance application. Thus, the SK we use shall be certifiable to SIL4 which corresponds to the assurance level required by the safety application.

For the security assurance of SK, we rely on Common Criteria (CC) [17] which is a well-established international standard for security evaluation and certification. The CC approach demands the developer of a product to conduct an asset identification, threat analysis describing countermeasures in the form of security objectives and deriving of the security functional requirements (SFRs) that fulfill the security objectives followed by the implementation, validation, and verification of the SFRs.

The IT life-cycle described in railway security prestandard DIN VDE V 0831-104 is comparable to these steps in CC. For instance, the procedure description matches how a CC security target for a product is developed.

For the CC threat analysis, the assets correspond to the resources (such as memory, CPU cores, files, interrupts, and CPU processing time) assigned to a partition and the applications belonging to different partitions are considered as attackers. The SFRs ensure that the confidentiality, integrity, and resource availability properties of the assets protected by the SK are preserved from an application belonging to a different partition according to the partitioning and information flow security policies. Thus, the SFR claims of the SK are highly relevant for its role in the MILS architecture, namely, providing mechanisms for separation and well-defined information flow channels between partitions. The detailed description of threat analysis, SFRs, and its validation and verification is described in the SK CC assurance artefacts.

DIN VDE V 0831-104 assumes that IT security products purchased on the market already have IEC 62443 compatible certification [2] and the ISASecure's interpretation of IEC 62443-4-1 suggests that COTS operating systems should at least be certified to CC EAL3 [18]. There are also ongoing discussions on how to combine CC and IEC 62443 (e.g., in the certMILS project [19]). The current understanding is that CC is used on subcomponents such as operating systems and IEC 62443-4-1 and IEC 62443-4-2 on integrated devices (components) designed to be used as part of the infrastructure (system) compliant to IEC 62443-3-2 and IEC 62443-3-3.

It is also important to highlight that assurance level (EAL) and SFRs are completely independent; i.e., there can be very weak security functionality and high assurance as well as very strong security functionality and assurance level that just fulfills the purpose. Moreover, from business point of view, the cost of evaluations grows exponentially [20] with higher EALs, and thus, any system designer would need to look for approaches to manage these costs by choosing the right assurance level. The French national security authority (ANSSI) also defines three levels of assurance with EAL3+ being a medium one [21].

Based on this state-of-the-art research, we have chosen the EAL3+ as the very minimum for the security assurance for the SK subcomponent while requiring very strong functional security requirements (i.e., separation) in the security targets of SK.

6.2. Security Applications for Integrity Protection and Resilience. In order to define our security architecture that fulfills the requirements defined in Section 4, we utilize the shell concept of Schlehuber et al. [22]. In the shell concept, safety is protected by a shell of security measures. Any communication with the safety part passes through the security shell. Thus, an attacker must successfully penetrate the security shell in order to attack the safety functionality.

The OCs are currently the weakest spot of the CCS architecture, as they are not protected by a surrounding building. Thus, we propose a security architecture for OCs in our system definition (see Figure 2). We call an OC that implements our proposed security architecture, as illustrated in Figure 4, a Secure Object Controller (SecOC). The depicted building blocks are discussed in this section. How

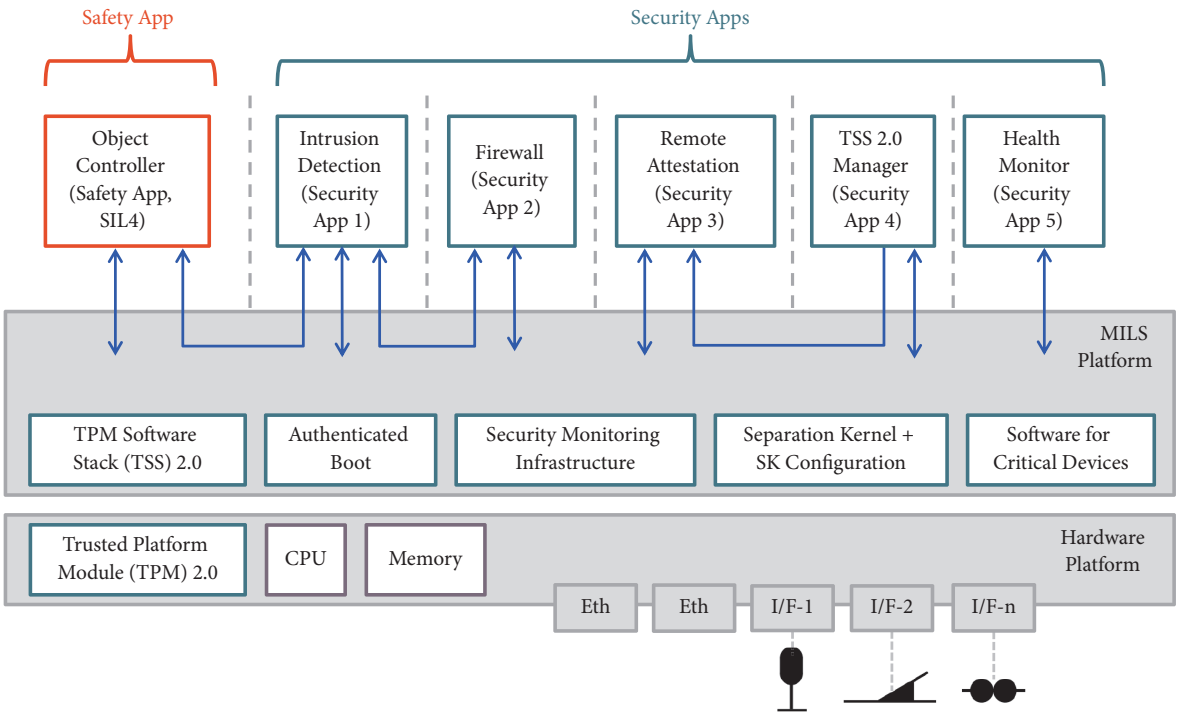


Figure 4: Hardware-based security architecture for the Secure Object Controller.

each building block contributes to the fulfillment of the security requirements is discussed in Section 7.

Hardware Platform. The SecOC is located remotely at railway tracks. To ensure that this safety-critical system is not manipulated by malicious attackers, a continuous proofing is performed by using trusted computing technologies [23]. As a hardware root of trust (ROT), a Trusted Platform Module (TPM) 2.0 specified by the Trusted Computing Group (TCG) [24] is used. It provides secure storage, secure execution for cryptographic operations, and additional features, e.g., facilitating authenticated boot and secure update procedures. The TPM is a special type of a hardware security module. In our case, it is implemented as a dedicated chip. Petri et al. showed that cryptographic agility, enhanced authorization, and modularity of the TPM 2.0 specification make it suitable to secure long-lived resource-constrained systems like controllers [25]. Authenticated boot and remote attestation build on the capability of the TPM to persistently store the status of the loaded software as so-called configuration measurements (hash-chains) within the TPM's Platform Configuration Registers (PCRs) [26].

To further protect the SecOC against physical attacks, the housing or chassis of the hardware platform raises an alert every time it is opened. The alerts are written into the TPM via the BIOS and can only be reset by authorized personnel. Chassis open alerts work even if the SecOC is disconnected from power supply and make physical interference from an attacker evident.

For the redundant network connection, the hardware platform provides two Ethernet interfaces. In order to steer the field elements, wired interfaces to signals, point machines, and train detection systems are designated.

Software (MILS) Platform. The software platform is constructed following the MILS methodology [14]. The security core of the proposed architecture builds on the SK. The MILS platform architectural layer provides a controlled interface to the hardware and enforces security policies for information flow, access control, and resource availability.

The TPM Software Stack (TSS) 2.0 specified by the TCG [24, 27, 28] provides an API for TPM 2.0 to applications and is also part of the platform. A software for critical devices controls direct access of hardware to the main memory without involvement of CPU and MMU that would otherwise bypass the SK. A Security Monitoring Infrastructure module inside the MILS platform collects information about system state and network traffic required for detecting the network anomalies and performing health monitoring.

Security Applications. Security functions necessary for the SecOC to fulfill the identified security requirements are realized by security applications (denoted as *Security App* in Figure 4) running on the MILS platform. This includes applications enabling the protection of the SecOC software integrity at boot- and runtime, integrity reporting, remote attestation, and secure software update as well as health monitoring, network traffic filtering, and intrusion detection (see Section 6.2).

Safety Applications. Alongside the security applications, one or more safety applications run on the MILS platform. In case

of the SecOC, we encapsulate the existing functionality of an OC (see Figure 1) in the safety application that operates a point or a signal.

6.2.1. Hardware Trust Anchor and Security Protocols. A TPM 2.0 provides a hardware root of trust (ROT). Using an authenticated boot procedure, the platform's ROT for Reporting generates a report about the boot process by creating evidence about the integrity of the booted software components. Using a remote attestation procedure, this evidence can be conveyed to an external verifier in order to appraise the evidence.

In our architecture, we will use the Time-Based Uni-Directional Attestation (TUDA) protocol [29] as remote attestation procedure. In contrast to traditional remote attestation procedures (e.g., [30]), the attestor (or the verifier, respectively, depending on which entity initiates the procedure) does not need to send a nonce, in order to prove the freshness of the created evidence, but uses trusted timestamps originating from an external trusted source of time. This approach has the advantage of decoupling the activities of creating the integrity evidence, conveying it and then appraising it via a verifier, therefore enabling the appraisal of past states and also reducing the utilization of the TPM significantly.

6.2.2. Authenticated Boot Security Architecture. In order to create evidence that enables integrity proving of a SecOC and its corresponding software components that compose the runtime environments, an authenticated boot procedure is introduced. Even before booting, integrity measurements—resilient hash-values of every software component involved, including the BIOS—are created and aggregated in a corresponding partition of the SecOC. Measurements of safety applications and their configuration data are created with the help of the introduced authenticated boot kernel component (see Figure 4) on startup and during observation of update processes. The separation enforced by the SK guarantees that the creation of integrity evidence, as a basis for remote attestation procedures, has no impact on the runtime behaviour of the safety applications.

The relationships of interfaces among the entities in the SecOC are illustrated in Figure 4. Basically, every higher level interface with respect to creation of evidence is based on the TSS 2.0. Secure communication channels between partitions are realized using communication objects that are governed by the SK. The SK provides the separation assurance required to shield the safety-related partitions from malicious or unintentional harmful use of these communication objects.

6.2.3. Runtime Integrity Monitoring. Considering that rebooting or updating an OC does not happen frequently, runtime mechanisms for integrity control and system resilience are required to fulfill the security requirements. This functionality is provided by a dedicated health monitor (see Figure 4) that continuously performs collection of data regarding application state, analyses this data for changes or anomalies and does reporting [31, 32]. For this purpose, the communication objects of the SK need to be extended with monitoring capabilities to detect failures or attacks in system safety or security services. Another data source for the health monitor is the kernel-level security monitor. The health monitor can be noninvasive or enable invoking specified resilience mechanisms similar to the ones proposed in [33, 34] desired for some particular scenarios. Thus, to reduce the service downtimes, a SecOC may be able to recover after failures or attacks, e.g., by automatically resuming to a known valid configuration instead of failing. In this case, this behaviour does not have any impact on the assurance level of the safety applications because the application itself is not changed by the health monitor.

6.2.4. Secure Update. Software, firmware, and configuration updates are necessary to fix software bugs and vulnerabilities in the SecOC. The new features introduced by the TPM 2.0 specification, such as monotonic counters and enhanced authorization, make it possible to protect system data, e.g., safety-relevant settings and to ensure that only updates from authorized sources are accepted and downgrade attacks are prevented [26].

6.2.5. Intrusion Detection. The Intrusion Detection System (IDS) provides a defense mechanism against network-based attacks and applies after the network traffic has been initially filtered by a firewall. Types of network traffic are distinct and invariable. Protocols and ports for control commands, secure update, and remote attestation are known in advance. The firewall can be configured by a predetermined whitelist. By collaborating among multiple SecOC instances in a defined area of the controlled field elements, the IDS is enabled to detect adverse commands, dangerous infrastructure configurations, and misuse on application level. Our IDS solution allows the SecOCs to validate received commands as an additional layer of defense beyond authenticated communication channels. Ability current OCs do not possess. The IDS is fine-tuned to the CI's network topology (i.e., the track layout and the position of signals and points) and the utilized transport and application protocols. In this way, the IDS is enabled to leverage context information of the controlled infrastructure to enhance the intrusion detection accuracy. In a second step, counteractions on detected intrusions are defined that respect the safety functions of the CI. We carefully design the intrusive counteractions such that they do not alter the network channel properties beyond the specification of latency and message loss that the safety application is anyway required to tolerate.

7. Coverage of Security Requirements

In the following, we argue that all security requirements defined in Section 4 are covered by the proposed hardware-based security architecture for the SecOC. Once a working prototype of our architecture is built, we evaluate it against the requirements with the proposed coverage.

R1 *The system shall detect unauthorized physical access to its subsystems and/or prevent relevant exploitations of physical access.*

Attack scenarios of physical access are addressed in several ways. General access to the chassis of the SecOC can be detected through the chassis open alert provided by the hardware platform (see Section 6.2). The access is reported using the authenticated TPM attestation means. Also the alteration of the main firmware of the device or the replacement of those hardware components that carry an Option-ROM can be detected by the attestation. Using the audit capabilities of the attestation protocol, even physical access to the device or management backend cannot alter postincident logs in order to, e.g., fake the firmware or configuration status of the device during an accident.

Additionally, the devices prevent the cloning of a devices identity, since the relevant data is stored inside the TPM. Though a TPM and the associated identity can be stolen, they cannot be duplicated. The use of monotonic counters inside the TPM also prevents so-called rollback attacks of local firmware, where older versions are used in place of more recent versions.

Please note that some physical manipulations are still possible, such as the tapping in between CPU and TPM. Even though concepts against such attacks exist (e.g., RayzorClam) they are not needed, since the same attacker class can reach results of direct I/O manipulation or hardware replacement much cheaper, than in-field PCB modifications.

R2 *The system shall not allow the compromising of a communication key.*

The TPM 2.0 provides a secure storage for communication keys and any other cryptographic keys as well as a secure execution environment for cryptographic operations to protect them from compromise.

R3 *The system shall not disclose classified or confidential data (such as access credentials) to any illegitimate user.*

Service access, user login, and privilege escalation events can be stored in the TPM in order to create evidence about system state that may enable unauthorized disclosure and can result in actions ranging from event notification to automated remediation or quarantine procedures. The TPM 2.0 provides a secure storage for cryptographic keys, a secure execution environment for cryptographic operations, and supports state-of-the-art cryptographic algorithms to protect confidential data from unauthorized disclosure.

R4 *The system shall exclude compromised endpoints from communication.*

Continuous proving of platform integrity and attestation allows detecting and excluding compromised endpoints from the communication. Using a suitable integrity measurement architecture policy, those attestations can validate the integrity of code as well as data elements of the platform. The proposed architecture will measure all complete static partitions, including code and data segments. Additionally, an interface exists, that these measured (and well-behaved) partitions will in turn use to measure their dynamic (runtime) configuration data.

The concept of measuring code as well as data and doing so context dependently is not unusual; the default Linux policy will measure all code (executables and libraries loaded by all users). For the root user, it will additionally measure all data files opened. For example, measured runtime data can be the state of the controlled field element (e.g., signal aspect and direction of the point). The states are known in advance so that reference measurements can be created.

R5 *The system shall not use insecure transfer methods.*

Our architecture does not use transfer protocols with known vulnerabilities. Update mechanisms are applied to react to newly discovered vulnerabilities (see R9).

R6 *The system shall not allow any unauthorized user to access an endpoint.*

Access to the endpoints is protected by user credentials.

R7 *The system shall not allow unauthorized and unauthenticated communication between endpoints.*

The TPM 2.0 can be used to equip each OC with a secure and unique identity that provides the basis for secure networking. Allowed network flows are assessed and enforced by a firewall.

R8 *The system shall not violate the runtime behaviour requirements.*

Separation mechanisms supported by the SK allow statically assigning user applications all resources needed for fulfilling its runtime requirements and provide guarantees that the safety application and security applications do not have to compete with each other for resources.

R9 *The system shall allow for the updating of security mechanisms, credentials, and configurations in order to patch known vulnerabilities.*

The TPM 2.0 based secure update mechanism enables remote updates for security mechanisms, credentials, and configurations and rollback protection.

R10 *The system shall not allow the execution of unauthorized software instances.*

Depending on the criticality of the software instance affected, an authenticated boot procedure enables countermeasures ranging from enabling restrictions on execution privileges to pausing affected partitions. A fine-granular action catalogue ensures that countermeasures cannot affect functions of a safety-critical partition.

R11 *The system shall maintain the transmission system requirements defined in EN 50159.*

An implementation of the architecture will be evaluated to assess that the requirements are not violated.

R12 *The system shall provide mechanisms to detect an undesirable system state change and anomalies.*

Based on the collected integrity evidence, changes in the boot system configuration can be revealed. The health monitor and the intrusion detection analyse the runtime behaviour of the applications and detects undesired states.

R13 *The system shall impede that an unauthorized user can force it into one of the fall-back levels defined by the railway safety process.*

The applications of the security part of the shell concept proposed by Schlehuber et al. [22] work free of interference with the applications in the safety part of the shell. Without substantial and well-defined reason, the security does not trigger fail-safe states. The security architecture provides resilience mechanisms that help to recover the system from attacks and reduce downtimes.

R14 *The system shall maintain the integrity of software, firmware, configuration, and hardware.*

Boot and runtime integrity control of software, firmware, and configuration together with resilience mechanisms help to protect the system from compromise. Depending on the hardware platform, the hardware integrity, a kernel interface conducting the respective measurements or platform certificates with hardware configuration attributes, is used to maintain the hardware integrity. Hardware integrity refers to the unmodified assembly of hardware to a composite device (i.e., the SecOC).

As our research is based on DIN VDE V 0831-104 which is a guideline to apply IEC 62443 to the railway domain, we believe that we sufficiently covered all security requirements. We are aware that it is hard to know when a list of security requirements is complete, as security—contrary to safety—is an iterative process. This is why our architecture contains the ability to be extended by further security measures if necessary.

8. Conclusion

In this paper, we report our practical experience obtained while executing a security requirements engineering process for railway signalling systems in a real-world scenario. This process is defined in the German prestandard DIN VDE V 0831-104 which must be conformed to in order to receive admission to operate railway infrastructure. Our reference architecture covers the components required to operate the track-side equipment necessary for safe train journeys, being signals and points. The reference architecture was translated by experts to a system definition featuring zones. Our risk analysis yielded a SL of 4 for the identified zones demanding the highest security requirements according to IEC 62443, a security standard for industrial control systems. Overall, the process is suited for finding useful system requirements, especially because it is constructed along already established safety and security standards. However, we came across some minor shortcomings that we presented in this paper. We proposed modifications to overcome them and analysed the impact of our suggestions on our real-world application.

The derived requirements were used to design a security architecture for safety-critical railway signalling networks based on a hardware platform allowing to run mixed-criticality applications. We propose the security architecture based on the MILS concept to comply with the identified security requirements. The concept is implemented using a separation kernel that controls resource sharing between safety and security applications. The security applications of our architecture employ health monitoring, authenticated boot, remote attestation procedures, secure updates, and anomaly detection services. These capabilities provide a wide spectrum of security-related information including complementary indicators that improve security assessments as well as cryptographic integrity evidence that can be conveyed to management systems to prove that a device is a trustworthy system. Additionally, secure audit logs are created that are able to document past states a device was in with the highest level of assurance. We analyse how the proposed security architecture fulfills the requirements we derived.

We will implement and demonstrate the feasibility of the proposed security architecture during the HASELNUSS project (https://haselnuss-projekt.de/index_en.php). The project aims to show that it is possible for safety and security applications to coexist in the safety environment in order to provide safe and secure railway operation. During the course of the project, we will evaluate our platform regarding the security requirements and work towards safety certification.

Data Availability

There is no underlying data to publish.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The work presented in this paper has been partly funded by the German Federal Ministry of Education and Research (BMBF) under the project "HASELNUSS: Hardwarebasierte Sicherheitsplattform für Eisenbahn-Leit- und Sicherungstechnik" (ID 16KIS0597K).

References

[1] S. Kriaa, L. Pietre-Cambacedes, M. Bouissou, and Y. Halgand, "A survey of approaches combining safety and security for industrial control systems," *Reliability Engineering & System Safety*, vol. 139, pp. 156–178, 2015.

[2] "DKE: Elektrische Bahn-Signalanlagen – Teil 104: Leitfaden für die IT- Sicherheit auf Grundlage der IEC 62443 (DIN VDE V 0831-104)," 2015.

[3] S. Islam, N. Suri, A. Balogh, G. Csertán, and A. Pataricza, "An optimization based design for integrated dependable real-time embedded systems," *Design Automation for Embedded Systems*, vol. 13, no. 4, pp. 245–285, 2009.

[4] J. Braband, "Towards an IT security risk assessment framework for rail- way automation," CoRR abs/1704.01175, http://arxiv.org/abs/1704.01175, 2017.

[5] M. Rocchetto and N. O. Tippenhauer, *On attacker models and profiles for cyber-physical systems*, Springer International Publishing, 2016.

[6] APTA Control and Communications Working Group, "Securing Control and Communications Systems in Rail Transit Environments – Part II: Defining a Security Zone Architecture for Rail Transit and Protecting Critical Zones," Tech. rep., APTA, 2013, http://www.apta.com/resources/standards/documents/apta-ss-ccs-rp-002-13.pdf, APTA-SS-CCS-RP-002-13.

[7] H. Bock, J. Braband, B. Milius, and H. Schäbe, "Towards an IT Security Protection Profile for Safety-Related Communication in Railway Automation," in *Computer Safety, Reliability, and Security*, pp. 137–148, Springer, 2012.

[8] S. Myagmar, A. J. Lee, and W. Yurcik, "Threat modeling as a basis for security requirements," in *Symposium on Requirements Engineering for Information Security (SREIS)*, vol. 2005, pp. 1–8, 2005.

[9] "CENELEC - European Committee for Electrotechnical Standardization: EN50128 - Railway applications — Communications, signalling and pro- cessing systems — Software for railway control and protection systems. No. EN 50128:2001 E, CENELEC Central Secretariat, rue de Stassart, 36, B- 1050 Brussels," 2010.

[10] S. Baruah, H. Li, and L. Stougie, "Towards the design of certifiable mixed- criticality systems," in *Proceedings of the Real-Time and Embedded Technology and Applications Symposium (RTAS), 2010 16th IEEE*, pp. 13–22, IEEE, 2010.

[11] J. A. Foss, P. W. Oman, C. Taylor, and W. S. Harrison, "The MILS architecture for high-assurance embedded systems," *International Journal of Embedded Systems*, vol. 2, no. 3/4, p. 239, 2006.

[12] K. Netkachova, K. Müller, M. Paulitsch, and R. Bloomfield, *Security-Informed Safety Case Approach to Analysing Mils Systems*, 2015.

[13] J. M. Rushby, "Design and Verification of Secure Systems," in *Proceedings of the Eighth ACM Symposium on Operating Systems Principles, SOSP '81*, pp. 12–21, ACM, New York, NY, USA, 1981.

[14] S. Tverdyshev, H. Blasum, B. Langenstein et al., "MILS Architecture," *EURO-MILS*, 2013.

[15] G. Heiser and K. Elphinstone, "L4 microkernels: The lessons from 20 years of research and deployment," *ACM Transactions on Computer Systems*, vol. 34, no. 1, pp. 1:1–1:29, 2016.

[16] G. Klein, J. Andronick, K. Elphinstone et al., "Comprehensive formal verification of an OS microkernel," in *ACM Transactions on Computer Systems*, vol. 32, pp. 2:1–2:70, 2014.

[17] "Common Criteria Sponsoring Organizations: Common criteria for information technology security evaluation. version 3.1, revision 5," http://www.commoncriteriaportal.org/cc/, 2017.

[18] ISA Security Compliance Institute, "SDLA-312 Security Development Life- cycle Assessment Version 3.0," http://www.isasecure.org/en-US/Certification/IEC-62443-SDLA-Certification, 2014.

[19] "certMILS project - H2020 project reference: 731456: certmils - compositional security certification for medium- to high-assurance cots-based systems in environments with emerging threats," http://www.certmils.eu/.

[20] G. C. Wilshusen, "Information Assurance: National Partnership Offers Ben- efits, but Faces Considerable Challenges," Tech. rep, United States Government Accountability Office, Washington, DC, USA, 2006, http://www.gao.gov/new.items/d06392.pdf.

[21] "ANSSI: Agence nationale de la sécurité des systèmes d'information: Anssi: Qualification de produit," https://www.ssi.gouv.fr/entreprise/qualifications/qualification-de-produit/.

[22] C. Schlehuber, M. Heinrich, T. Vateva-Gurova, S. Katzenbeisser, and N. Suri, "Challenges and approaches in securing safety-relevant rail- way signalling," in *Proceedings of the Security and Privacy Workshops (EuroS&PW) 2017 IEEE European Symposium on*, pp. 139–145, IEEE, 2017.

[23] N. Asokan, J. Ekberg, K. Kostiainen et al., "Mobile Trusted Computing," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1189–1206, 2014.

[24] Trusted Computing Group, *Trusted Platform Module Library Specification, Family 2.0, Level 00, Revision 01.38 edn*, 2016.

[25] R. Petri, M. Springer, D. Zelle et al., "Evaluation of lightweight TPMs for automotive software updates over the air," in *Proceedings of the 4th Embedded Security in Cars (escar)*, USA, 2016.

[26] A. Fuchs, C. Krauß, and J. Repp, "Advanced Remote Firmware Upgrades Using TPM 2.0," in *Proceedings of the 31st International Conference on Information Se- curity and Privacy Protection IFIP SEC 2016*, pp. 276–289, Springer International Publishing, 2016.

[27] "TSS system level API and TPM command transmission interface specification," http://www.trustedcomputinggroup.org/resources/tss_system_level_api_and_tpm_command_transmission_interface_specification, 2016.

[28] "TSS TAB and resource manager," http://www.trustedcomputinggroup.org/resources/tss_tab_and_resource_manager, 2016.

[29] A. Fuchs, H. Birkholz, I. McDonald, and C. Bormann, "Time-Based Uni- Directional Attestation," in *Internet-Draft draft-birkholz-i2nsf-tuda, The Inter- net Engineering Task Force (IETF)*, 2017, https://datatracker.ietf.org/doc/draft-birkholz-i2nsf-tuda/.

[30] R. V. Steiner and E. Lupu, "Attestation in wireless sensor networks: A survey," *ACM Computing Surveys*, vol. 49, no. 3, pp. 51:1–51:31, 2016.

[31] A. M. Azab, P. Ning, E. C. Sezer, and X. Zhang, "HIMA: A hypervisor- based integrity measurement agent," in *Proceedings of the Computer Security Applications Conference, 2009. ACSAC '09. Annual*, pp. 461–470, 2009.

[32] B. D. Payne, M. Carbone, and W. Lee, "Secure and flexible monitoring of virtual machines," in *Proceedings of the 23rd Annual Computer Security Applications Conference*, ACSAC, 2007.

[33] M. Dinkel, S. Stesny, and U. Baumgarten, "Interactive self-healing for black-box components in distributed embedded environments," in *Proceedings of the Communication in Distributed Systems (KiVS), 2007 ITG-GI Conference*, pp. 1–12, 2007.

[34] D. Garlan, S. Cheng, A. Huang, B. Schmerl, and P. Steenkiste, "Rainbow: architecture-based self-adaptation with reusable infrastructure," *The Computer Journal*, vol. 37, no. 10, pp. 46–54, 2004.

Hindawi
Security and Communication Networks
Volume 2019, Article ID 4869167, 11 pages
https://doi.org/10.1155/2019/4869167

Research Article

SSPSoC: A Secure SDN-Based Protocol over MPSoC

Soultana Ellinidou, Gaurav Sharma, Théo Rigas, Tristan Vanspouwen, Olivier Markowitch, and Jean-Michel Dricot

Cybersecurity Research Center, Université Libre de Bruxelles, Belgium

Correspondence should be addressed to Soultana Ellinidou; soultana.ellinidou@ulb.ac.be

Received 7 July 2018; Revised 14 December 2018; Accepted 6 February 2019; Published 18 March 2019

Guest Editor: Daniel Schneider

Copyright © 2019 Soultana Ellinidou et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In recent years, Multi-Processor System-on-Chips (MPSoCs) are widely deployed in safety-critical embedded systems. The Cloud-of-Chips (CoC) is a scalable MPSoC architecture comprised of a large number of interconnected Integrated Circuits (IC) and Processing Clusters (PC) destined for critical systems. While many researches have focused on addressing the hardware issues of MPSoCs, the communication over them has not been very well explored. Following the SDN concept, we propose a new protocol in order to secure the communication and efficiently manage the routing within the CoC. The SSPSoC includes a private key derivation phase, a group key agreement (GKA) phase, and a data exchange phase in order to ensure that basic security primitives are preserved and provide secure communication. Furthermore, a network of 1-30 nodes is set in order to validate the proposed protocol and measure the network performance and memory consumption of the proposed protocol.

1. Introduction

Since the late 1990s the rise of System-on-Chips (SoCs) has caused a huge technological progress with a dramatic involvement not only on the field of electronics but also on Internet of Things (IoT) and Internet of Everything (IoE) field [1]. IoT and IoE brought into the surface a wide variety of applications, able to satisfy our needs in transportation, health-care, manufacturing, and energy management with diverse requirements, which traditional SoCs are not always able to support. Hence, the creation of a flexible, technology aware SoC design is vital.

A new scalable MPSoC architecture, called Cloud-of-Chips (CoC) (Figure 1), could be a possible solution, which consists of a combination of multiple Integrated Circuits (ICs) and IC building blocks interconnected together with different communication speeds and hierarchy levels. The CoC design contains a Printed Circuit Board (PCB) of multiple ICs where each IC contains scalable Processing Clusters (PCs). Each PC comprises a combination of High Performances (HP) and Low Power (LP) cores and thus enables a heterogeneous system architecture [2]. The on-chip data communication is managed by Network on Chip (NoC) [3], which supports regular interconnected topologies, such as MESH, TORUS, etc. [4].

In order to manage the routing on CoC with multiple ICs, the size of routing tables will be large enough not to be accommodated on ordinary NoC routers. The memory overhead for routing tables will grow by $n^2 * k$ units where n^2 is the number of PCs on each IC and k is the number of IC on each PCB. Therefore, in order to achieve secure communication among PCs and ICs on CoC but also to manage the traffic of the network with an efficient manner, the Software Defined Networking (SDN) technology [5] could be adopted. Hence the design of a secure networking approach of SDN technology and its integration into hardware by moving network intelligence and services from complex and expensive physical switches and routers and placing them into software that runs on the top of the underlying hardware need to be addressed.

The SDN-based networking strategy enables the communication between any two PCs on CoC. There is no existing literature for CoC. However, there is limited literature available in NoC-based MPSoC which includes SDN as a packet routing approach. SDNoC is a NoC communication paradigm rather than a specific design and implementation,

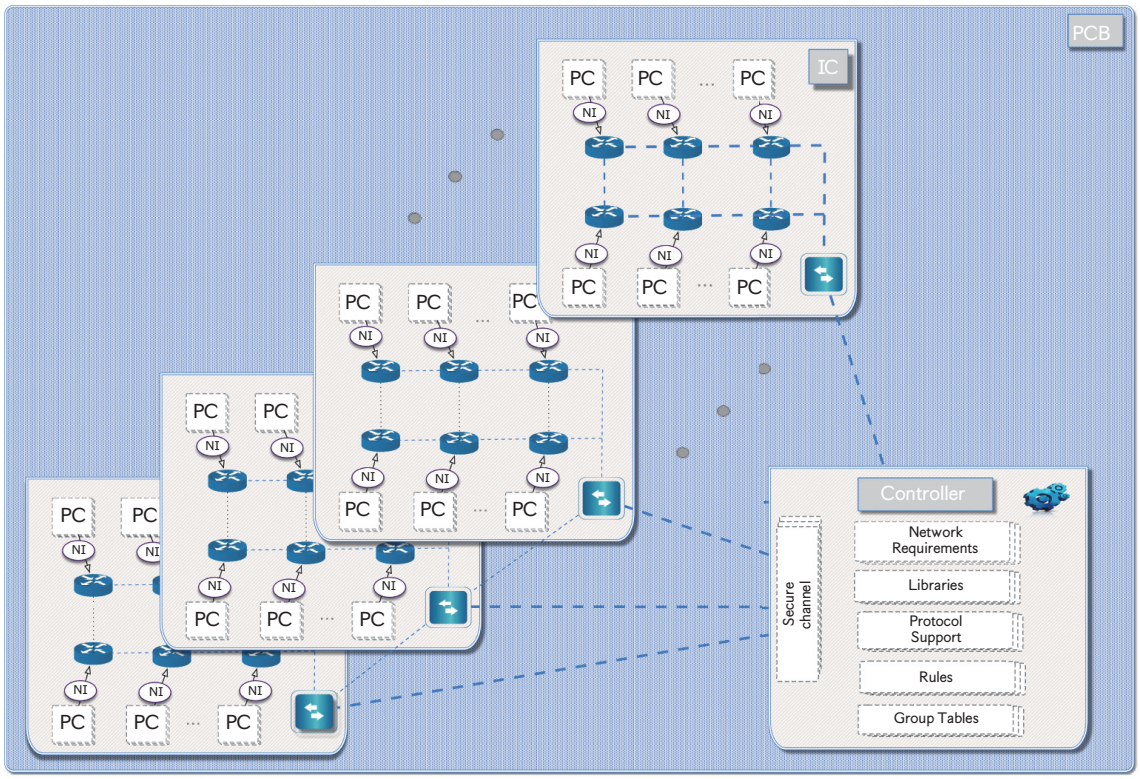


Figure 1: CoC architecture.

presented for first time in 2014 [6]. Specifically Cong et al. propose a SDNoC architecture where the control plane is deployed as a distributed unity at each router; however this is contrary to SDN philosophy because both planes are placed inside the router. Afterwards Sandoval et al. [7, 8] propose an SDNoC architecture which consists of three layers: Operating System, Network Operating System, and Infrastructure. In this research the authors made an assumption that the routers need configuration data by the SDN controller. However flows, that are not managed by the SDN controller, use the XY routing algorithm. Thereafter Scionti et al. [9] use the SDN architecture in order to explore dynamic changes in the network topology; each Processing Element (PE) has specific instructions to control the network topology by software, including switch off the links which are not used. Another interesting approach is proposed by Berestizshevsky et al. [10], in which an SDNoC strategy by implementing a network manager is introduced, acting as controller with global view of the network, which controls the network in adaptive manner. However in order to be deployed on chip, a detailed API implementation and standardization between control plane and data forwarding plane need to be considered.

In CoC, the network is separated into two main abstraction hardware levels: IC and PCB. On the IC level the hardware routers on Network on Chip (NoC) are functional as SDN switches and the routing on individual IC is managed by an IC level controller. For the inter IC communication, a SDN switch is placed at the boundary of each IC and it is connected to a PCB level controller [11] through physical links. The controller consists of a *Secure Channel*, where the private and public keys are stored for the communication between SDN switches, *Rules* where the applicable rules to the switches are included, *Protocol Support* where the supported communication protocols are included, and *Libraries* and *Network Requirements*, where the network interface (NI) informs the controller about the traffic of the network and according to that field the *Rules* is constructed (Figure 1).

There are several SDN-based communication protocols seen in literature but the most used is the OpenFlow (OF). OF establishes a unicast communication channel between each individual switch and the controller. It allows the controller to discover OF-compatible switches, create rules for the switching hardware, and also collects statistics [12]. This protocol has several security flaws that can be exploited to compromise the network. Unfortunately, the SDN paradigm is susceptible to several security breaches [13, 14]. However, in this paper, the main concern is the communication security among SDN switches (on each IC) and controller. In addition to the switch controller unicast communication, multicast communication is also needed to address the issue of secure transfer of routing updates to all/some of the switches [15, 16].

The existing security solution such as the Transport Layer Security (TLS) protocol is not well enforced in the current version of the OF standard [17]. The specification mentioned that "the switch initiates a standard TLS or TCP connection to the controller" which means that using TLS is completely optional. Moreover, the Public Key Infrastructure (PKI) overhead includes generation and signing of digital

certificates for switches and controller. This makes PKI based solution less acceptable for and MPSoC architecture. Therefore, we propose a more suitable solution that fits to unique characteristics of the CoC architecture.

In SDN concept, multiple vulnerability analyses have been performed [3, 5, 7–9, 12, 13, 18–22]; however, none of them attempts to extensively analyze the security issues of the SDN architecture. In the latest version of OpenFlow, in order to provide authentication and encryption of the connection between switch and controller, TLS is suggested security mechanism [23]. However, the latest OpenFow Switch specification [17], presented by Open Networking Foundation (ONF), does not provide enough information about security features and the implementation of TLS. The lack of TLS use could lead to fraudulent data insertion and DoS attacks [24]. Under the umbrella of Public Key Cryptography (PKC), TLS protocol requires a Certificate Authority (CA) to generate the CA's key, certificates for the controllers, switches, and then the signing of these certificates with the CA's key. Afterwards the certificates and the keys of the network entities will be deployed to the respective devices.

1.1. Group Key Agreement. The existing literature on SDN security refers to point-to-point communication between switches and controller. However, running different applications on CoC architecture creates multiple routing paths among ICs and therefore, multiple switches interacting with SDN controller quite frequently. Generally, the application based logical subsystems will be created which may involve multiple ICs (with different components such as GPU, crypto processor, etc.). Our idea is to create a secure point-to-multipoint communication between controller and group of switches, i.e., secure multicast. Therefore, two group key agreement (GKA) protocols are chosen for our architecture: the Sharma et al. approach [25] and the Teng et al. approach [26]. The protocols that we chose are balanced and imbalanced group key agreement (GKA) protocols. In balanced GKA protocols, all the participating nodes share the same computation burden while, in imbalanced, the powerful node (in our case, the controller) is majorly responsible for expensive computations. Sharma's protocol comes with the benefit of achieving mutual authentication using signature and of course the assurance that every participant at the end is in possession of the valid group key. However, the Teng protocol does not provide mutual authentication, thus trading security for efficiency.

1.2. Contribution. The need of secure communication between different hierarchy layers on MPSoC architecture in conjunction with the necessity of low power computation, scalability, and flexibility not only between the network nodes but also in the network as a whole leads us to the structure of a new communication protocol. This protocol will provide authenticity to network entities, low power consumption for the system, and a new message stack in order to have a quick secure communication among switches and controller.

The contribution of this paper can be summarized as follows:

(i) The design of new SDN-based protocol, which has three main functionalities: the derivation of keys for every node in our network through private key generator (PKG), the establishment of a secure group of participants, and the secure communication between the participants.

(ii) The validation of the proposed protocol, which is based on the SDN concept (by using switches and controller as network entities).

(iii) The performance analysis of two GKA protocols in order to verify which is more suitable in order to cover the second functionality of the proposed protocol in the view of running time and memory consumption.

The rest of the paper is organized as follows: in Section 2 the security requirements are presented; based on them it follows Section 3, where the group key agreement approach is analyzed. Afterwards, in Section 4, the SSPSoC communication protocol for an SDN-CoC network is described, where the architecture, the packet format, the proposed network messages, and the three different phases of the SSPSoC protocol, Private Key Extraction, Group Formation, and Switch Controller Communication, are presented. It follows the validation and the performance analysis of the proposed protocol in Section 5. Last but not least our research is summarized in the last section.

2. Security Requirements

The CoC architecture contains multiple switches and a single controller to manage the overall communication among ICs. The infrastructure to implement identity based cryptography requires an on-board PKG. The overall communication security on this layer (switch controller) can be investigated from two viewpoints. The first view is to securely deliver the private keys to switches and controller. The second view covers the secure communication among all the switches and the controller. We achieve this security using an authenticated group key agreement protocol.

2.1. Phase 1. The foremost issue to address is to transport the private key and required security parameters to all the switches and the controller. The PKG generates the private key for all switches and the controller and delivers it securely. The literature refers to using a secure channel but they do not specify exactly what this channel could be and its security requirements. The possible threats and solutions are as follows:

(i) In order to ensure that the only legitimate nodes can receive identity ID and private key, node authentication must be performed by the *PKG*.

(ii) A counterfeit PKG with different master key generates private keys and IDs for the nodes. This *PKG* is able to decrypt all the traffic between nodes and controller.

In this case, authentication of the PKG by the nodes is also needed.

(iii) An attacker can eavesdrop on the response of the PKG and steal the private key of a node. A solution must be there to ensure the confidentiality of communication between a node and the PKG.

(iv) An attacker can sniff the packets exchanged between a node and the PKG and replay them later to obtain a private key.

(v) An attacker can manage to compromise the integrity of the packets between node and PKG.

2.2. Phase 2. This phase refers to switch controller group communication where we adopted a GKA protocol to secure it. The common threats are spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privileges. The authenticated GKA protocol provides authentication of all participants. As the session key is derived, rest of the communication is encrypted using AES-GCM with session key. Therefore, confidentiality, authenticity, integrity, and nonrepudiation are maintained. To address denial of service and authorization issues, separate precautions need to be enabled.

3. Group Key Agreement

3.1. Assumptions. Before the design of the protocol some assumptions were vital to be made:

(i) The network consists of multiple nodes, which can be either a controller or SDN switches or the PKG, which derives the private keys to the network entities.

(ii) The private ID-based keys are provided to the participants of the group by PKG, which is the private key generator. Suppose that *msk* is the master secret key of the PKG and $KPub_{PKG}$ is the public key.

(iii) Given that there are n entities in the network, $U_1, U_2, \ldots, U_n$ is a group of participants in a GKA session for establishing a group key. Assuming that U_n will be the controller of every group, there are $n-1$ nodes (SDN switches) in this group.

3.2. GKA Protocols. By taking into account the above assumptions and the security requirements, we present the two GKA protocols that we based on and we constructed our protocol. The two protocols consist of 3 major phases: The *Setup* phase (hash functions, group generators, and pairing), the *Key Extraction* phase (nodes and controller obtain their private keys from PKG), and the *Key Agreement* phase (establishment of a group session key for participants).

The Teng protocol [26]:

Setup

(i) The PKG chooses two groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q, a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \longrightarrow \mathbb{G}_2$.

(ii) The PKG selects two random generators P and Q of $\mathbb{G}_1$.

(iii) The PKG selects $s \in \mathbb{Z}_q^*$ as the *msk* and sets $KPub_{PKG} = sP$.

Private Key Extraction. Defining as input parameters, *msk* and $ID_i \in \{0,1\}^*$ with ID_i being the ID of the node,

(i) The PKG computes $KPriv_i$ as $S_i = (q_i + s)^{-1}Q$ where $q_i = H(ID_i)$.

(ii) The PKG communicates secretly $KPriv_i$ to node i.

(iii) The public key of node i is $T_i = q_iP + KPub_{PKG} = (q_i + s)P$.

Key Agreement Round 1. Each participant

(i) U_i selects a random $r_i \in Z_q^*$.

(ii) U_i precomputes $P_i = r_iT_i$.

(iii) U_i $(1 \le i \le n)$ sends P_i to the controller S.

Key Agreement Round 2. Upon receiving P_i from all nodes, each participant

(i) U_i $(1 \le i \le n)$, S chooses random $r \in Z_q^*$.

(ii) U_i computes $Q_i = rP_i$.

(iii) U_i broadcasts Q_i $(1 \le i \le n)$, keeping r secret.

Key Computation. On receiving Q_j $(1 \le i \le n)$,

(i) U_i computes the final session key as

$$\begin{aligned} sk &= e\left(Q_i, S_i\right)^{r_i^{-1}} e\left(Q_1 + Q_2 + \cdots + Q_n, Q\right) \\ &= e\left(P, Q\right)^{r + rr_1(s+q_1) + \cdots + rr_n(s+q_n)}. \end{aligned} \tag{1}$$

Precomputation. The following tuples (r_i, r_i^{-1}, P_i) should be created and stored in the memory storage of the nodes before the execution of the GKA. This essentially reduces the computation cost of the first round for the nodes and also improves the speed of key computation phase.

The Sharma protocol [25]:

Assumption. Let *pid* be the set of the identities of the participants in one session of the protocol and *sid* the session identifier.

Setup

(i) The PKG selects an EC group G of prime order q. Let P be a generator of group G.

(ii) The PKG computes the system's public key as $KPub_{PKG} = sP$ by choosing a master secret $s \in \mathbb{Z}_q^*$.

(iii) The PKG chooses cryptographic hash functions $H_1 : \{0,1\}^* \times G \longrightarrow \mathbb{Z}_q^*$, $H_2 : \{0,1\} \times G \times G \longrightarrow \mathbb{Z}_q^*$ and $H : \{0,1\}^* \longrightarrow \{0,1\}^k$.

Key Extraction. Defying the system parameters $Params = \{G, q, H_1, H_2, H_3, H, KPub_{PKG}\}$ and by keeping the master key secret,

(i) The PKG selects $r_i \xleftarrow{\$} \mathbb{Z}_q^*$ and computes $R_i = r_i P$.

(ii) The PKG computes the private key for the user U_i as $KPriv_i = r_i + sH_1(ID_i, R_i)$.

(iii) Each participant U_i can verify the private key as $KPriv_i P = R_i + H_1(ID_i, R_i)KPub_{PKG}$.

Key Agreement Round 1. Each participant

(i) U_i $(1 \le i \le n)$ chooses $eph_i \xleftarrow{\$} \mathbb{Z}_q^*$ and computes $l_i = H_3(eph_i, KPriv_i)$ and $L_i = l_i P$.

(ii) U_i $(1 \le i \le n)$ selects a random string $k_i \in \{0,1\}^k$. Each user, except U_n computes $H(k_i)$. The user U_n masks the randomness as $\widetilde{k}_n = H(k_n, x_n)$ where x_n is the long-term secret of U_n.

(iii) U_i $(1 \le i \le n)$ computes $H(\widetilde{k}_n)$.

(iv) U_i $(1 \le i \le n)$ broadcasts the tuple $\langle L_i, H(k_i), H(\widetilde{k}_n), R_i \rangle$ to all $n-1$ members.

Key Agreement Round 2. Upon receiving the message $\langle L_j, H(k_j), H(\widetilde{k}_n), R_j \rangle$, each participant

(i) U_i computes $U_{ij} = l_i L_j$ and $L = L_1 \parallel L_2 \parallel .. \parallel L_n$.

(ii) U_i, except U_n, computes $K_{ij} = H(U_{ij}) \oplus k_i$. the user U_n computes $mask = H(U_{ij}) \oplus \widetilde{k}_n$.

(iii) U_i chooses another random number $t_i \in \mathbb{Z}_q^*$ and computes $T_i = t_i l_i P$. Also computes the signature on $\langle L, T_i \rangle$ as $\sigma_i = t_i l_i + KPriv_i H_2(ID_i, L, T_i, pid)$.

(iv) U_i broadcasts $\langle K_{ij}\ (1 \le j \le n, j \ne i), mask, \sigma_i, T_i \rangle$ to all $n-1$ members.

Key Computation. Upon receiving $\langle K_{ji}, mask, \sigma_i, T_i \rangle$, each participant

(i) U_i verifies the received signature as $\sigma_i P = T_i + (R_i + H_1(ID_i, R_i)KPub_{PKG})H_2(ID_i, L, T_i, pid)$.

(ii) U_i computes $\widetilde{k}_j = H(U_{ji}) \oplus K_{ji}$. (Similarly, $\widetilde{k}_n$ can be computed using mask.)

(iii) Note that $U_{ij} = l_i L_j = l_i l_j P = l_j l_i P = l_j L_i = U_{ji}$.

(iv) U_i checks the k_i as $H(k_j) = H(\widetilde{k}_j)$ for $(1 \le j \le n, j \ne i)$.

(v) U_i computes the session identity $sid = H(k_i) \parallel H(k_2) \parallel \ldots \parallel H(\widetilde{k}_n)$.

(vi) The session key is computed as $sk = H(k_1 \parallel k_2 \parallel \ldots \parallel \widetilde{k}_n \parallel sid \parallel pid)$.

4. Communication Protocol

In this section, the network architecture following the packet format, the network messages that are broadcasted within our network, and the three phases of the proposed protocol are introduced.

4.1. Architecture. The proposed architecture of the full system is presented on Figure 1. However our network architecture consists of 3 main network entities:

(i) A PKG, which is considered as a trusted third party and generates the corresponding private key to the rest of the nodes (switches and controller).

(ii) A centralized controller with a broader network view to manage the routing of the packets within the network.

(iii) Multiple switches which are responsible to route the packets between the PC

The communication between controller and switches is managed through a virtual network running on the top of the CoC.

4.2. Packet Format. The packet format is the core of the protocol stack. Every packet consists of a header structure, which is 32-bit long (Figure 2) [11]. The header message format consists of three main fields. Firstly, the version field indicates the version of communication protocol to which this message belongs. Secondly, the length field indicates where this message will end in the byte stream starting from the first byte of the header. Thirdly, the xid, or transaction identifier, is a unique value used to match requests to responses. The type field which indicates what type of message is present and how to interpret the payload is version dependent and we can see the messages that it is including above. Furthermore every message that travels across the network consists of the same header of 32 bits. However the payload size depends on the length field that is provided through header message and it can vary according to the type of the message. Afterwards it included the source and destination ID following the addressing format that we previously presented. Another field is the type of the packet; for example, it could be opcodes for a given processor, followed by some padding and the data. As far as the messages of the type field, a specific message stack is designed, which is presented afterwards.

4.3. Network Messages. The different types of messages, which were designed and integrated into packet format, are depicted in Table 1. The SSPSoC protocol includes 8 types of messages with different content. These messages are flowing through the links between the network entities. In addition the type value of the messages is used to distinguish GKA protocol messages from other messages that might be circulating on the network and one byte is used to encode the message type.

4.4. SSPSoC Network Initialization

Phase 1 (obtain private key). During the first phase, the switches and the controller communicate with the PKG, in order to obtain their long-term private keys. One of the major issues of concern on this phase was the security level of the communication between the nodes and the *PKG*, by

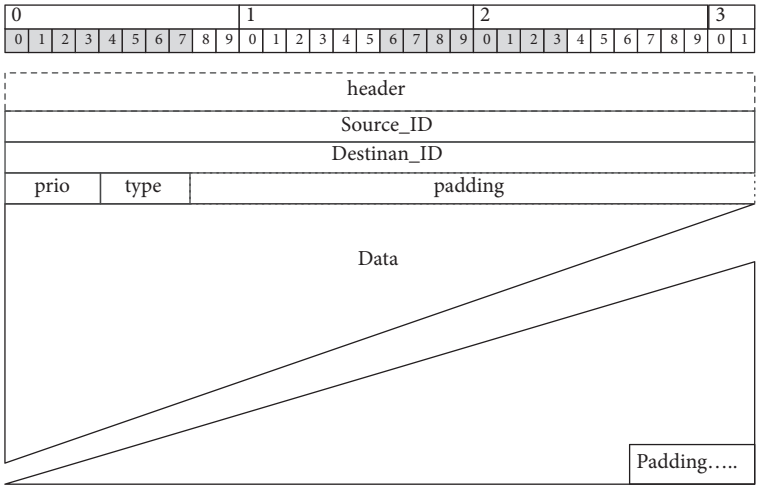


Figure 2: Packet format [11].

Table 1: Designed network messages.

Type	Type Value	Description	Contents
KEY_REQUEST	0x06	Sent by nodes to the PKG	Enc(timestamp), IV, tag
KEY_REPLY	0x07	Sent by PKG to nodes as a reply to KEY_REQUEST message	Enc(System Parameters, node ID, private key), IV, tag
JOIN	0x01	Broadcasted by nodes who wants to join a group	node ID, timestamp, JOIN token
INVITE	0x02	Broadcasted by controller for inviting nodes to form a group	(participant ID_1, node $ID_1, \ldots$, participant ID_n, $Node\ ID_n$)
READY	0x03	Broadcasted by nodes as a reply to INVITE message	participant ID, timestamp, READY token
ROUND_1	0x04	Contains cryptographic material for the first round of the GKA	sender ID, Crypto R1
ROUND_2	0x05	Contains cryptographic material for the second round of the GKA	sender ID, receiver ID, Crypto R2
DATA	0x08	Contains encrypted data with the group key	sender ID, receiver ID, Enc(data), IV, tag

tackling the problem of establishing a secure channel for the private key transmission. However, keeping the private key confidential is not the only security consideration; we should also take into account the authentication of the nodes. The authentication will ensure that only legitimate nodes can obtain a private key from the *PKG*. For this reason, the implementation of authenticated KEY_REQUEST messages is mainly used.

A node first determines a timestamp (t_s) to prevent the replay attacks [27]. Afterwards it generates the random part of the Initialization Vector (IV) and it encrypts t_s using the Preshared Symmetric Secret Key (PSK) and AES in Galois Counter Mode (AES-GCM) [28]. AES-GCM outputs the ciphertext c and the authentication tag: $c, tag = AES_{PSK,IV}(t_s)$. Thereafter the node sends a KEY_REQUEST message to the PKG, which contains the IV, the ciphertext, and the authentication tag. It follows a process where the *PKG* decrypts the ciphertext and checks the authentication tag. If the tags are matching, it will check that the decrypted timestamp is within a given threshold. In case the timestamp is valid, it will generate a random node ID and it will extract the associated private key, $KPriv(i)$. Thereupon it generates a random IV and encrypts them using the PSK and AES-GCM and sends the IV, ciphertext, and the authentication tag to the node. The steps of the this procedure are depicted on Figure 3.

Phase 2 (form a group). On this phase each switch communicates with the controller in order to show interest to join a group. The controller decides upon the group members and invites them to join the group.

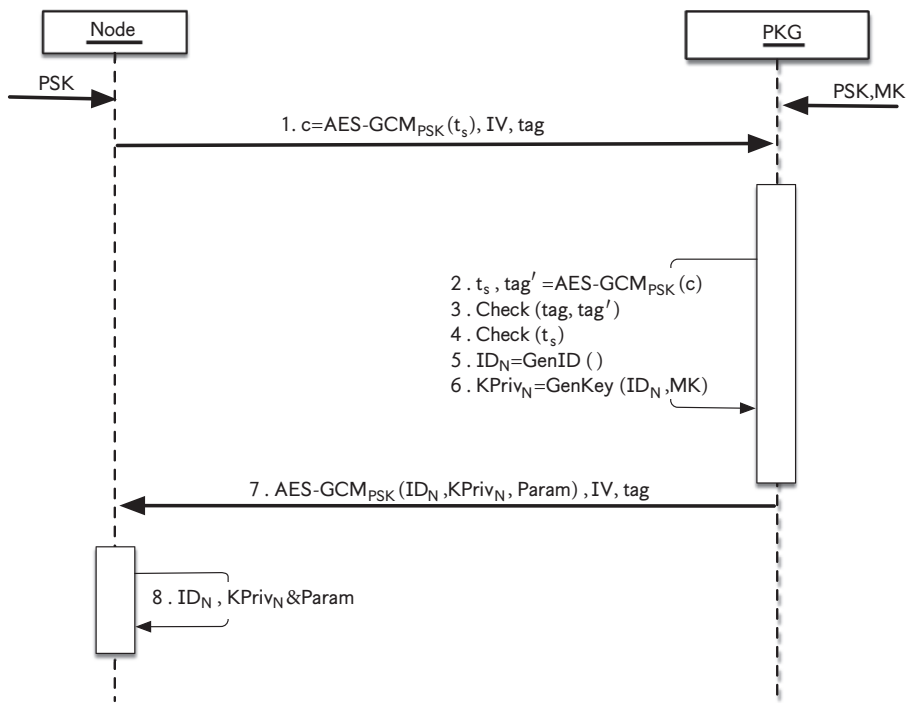


Figure 3: Private key exchange.

Firstly we are assuming that Phase 1 has been already performed and that the switches and controller have securely obtained their private keys. The controller has the power to decide which participants to invite to join a group, according the *Network Requirements* and *Rules* that includes (Figure 1). However when a switch wants to join an already existed group, it broadcasts a `JOIN` message with its node ID, without knowing the ID of controller. The controller is waiting for `JOIN` messages in order to start forming a group of switches. The behavior of the controller when receives `JOIN` message depends on the characteristics and requirements of the running applications which are translated and stored on the *Network Requirements* and *Rules* of the controller. As soon as the controller receives a number of `JOIN` messages and a group has been created, it broadcasts an `INVITE` message to all participated switches of the group, including a given session participant *ID* and node *ID* of all members of the group. The switches afterwards verify that they received the invitation and it follows the group key agreement process, where they are performing two rounds of messages. The two rounds are described in Section 3.

Phase 3 (switch controller communication). Once the group has been formed, we move into the last phase which is used for data exchange. In this phase the controller exchanges data messages with the groups of switches. Furthermore before the controller start exchanging any message with a group of switches, it checks the *Group Table* where all the IDs of group participants information that we described in the previous phase are stored. In case of data transmission between a group of switches and controller, the controller is using the *Secure Channel* where it encrypts the data. The controller encrypts the data using AES in GSM mode, the group key, and an *IV*.

5. Validation and Performance Analysis

As far as the performance analysis of SSPSoC protocol, we based on a simple scenario with three participants: PKG, Switch, and Controller. The messages that will be exchanged between three participants are depicted in Figure 4. As a first step the switch and controller will obtain a private key from PKG by establishing a TCP connection and transmitting the `KEY_REQUEST` message, the PKG will reply by `KEY_REPLY` message. While a TCP connection is needed in order to conduct a validation of our protocol, Layer 4 headers and protocols are not needed in the context on MPSoC, thus before its integration into an MPSoC platform some proper modifications should be performed. Afterwards we move into GKA process, where we implemented the Sharma protocol and Teng protocol, described in Section 3. In order to fit these two GKA protocols in our scenario we implement the following steps:

(1) A switch broadcasts a `JOIN` message, which contains its *ID*; the destination is always the controller and waits for an `INVITE` message.

(2) The controller receives the `JOIN` message, makes a decision about the participants of the group, and broadcasts an `INVITE` message to them, which contains the *IDs* of all the invited participants and waits for `READY` messages.

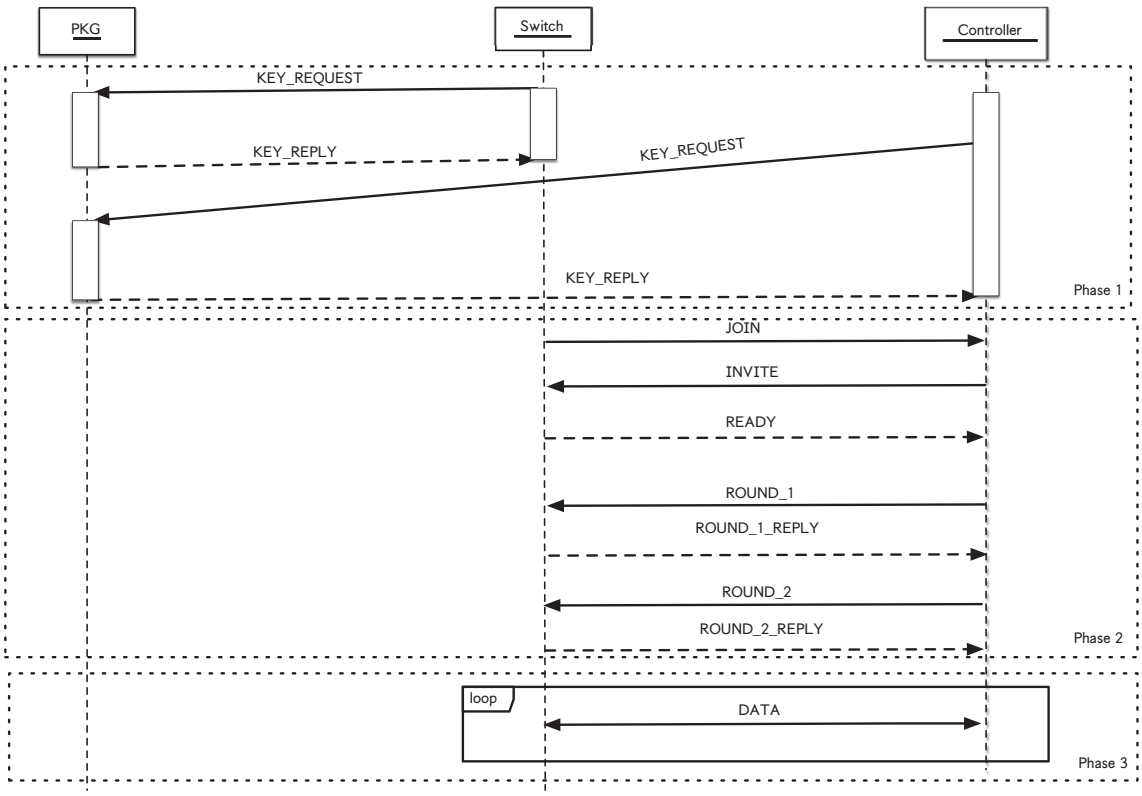


Figure 4: SSPSoC message layer.

(3) The switch receives the `INVITE` message and creates a list of participants. If the *ID* is valid, based on the list, it broadcasts a `READY` message.

(4) The controller remains in idle mode until it receives `READY` messages from the switches that are participants of the group for a specific time. Afterwards it sends a `ROUND_1` message and waits for `ROUND_1_REPLY` messages.

(5) As soon as the switch receives the `ROUND1` message, it broadcasts a `ROUND_1_REPLY` message by waiting for `ROUND_1` and `ROUND_2` messages.

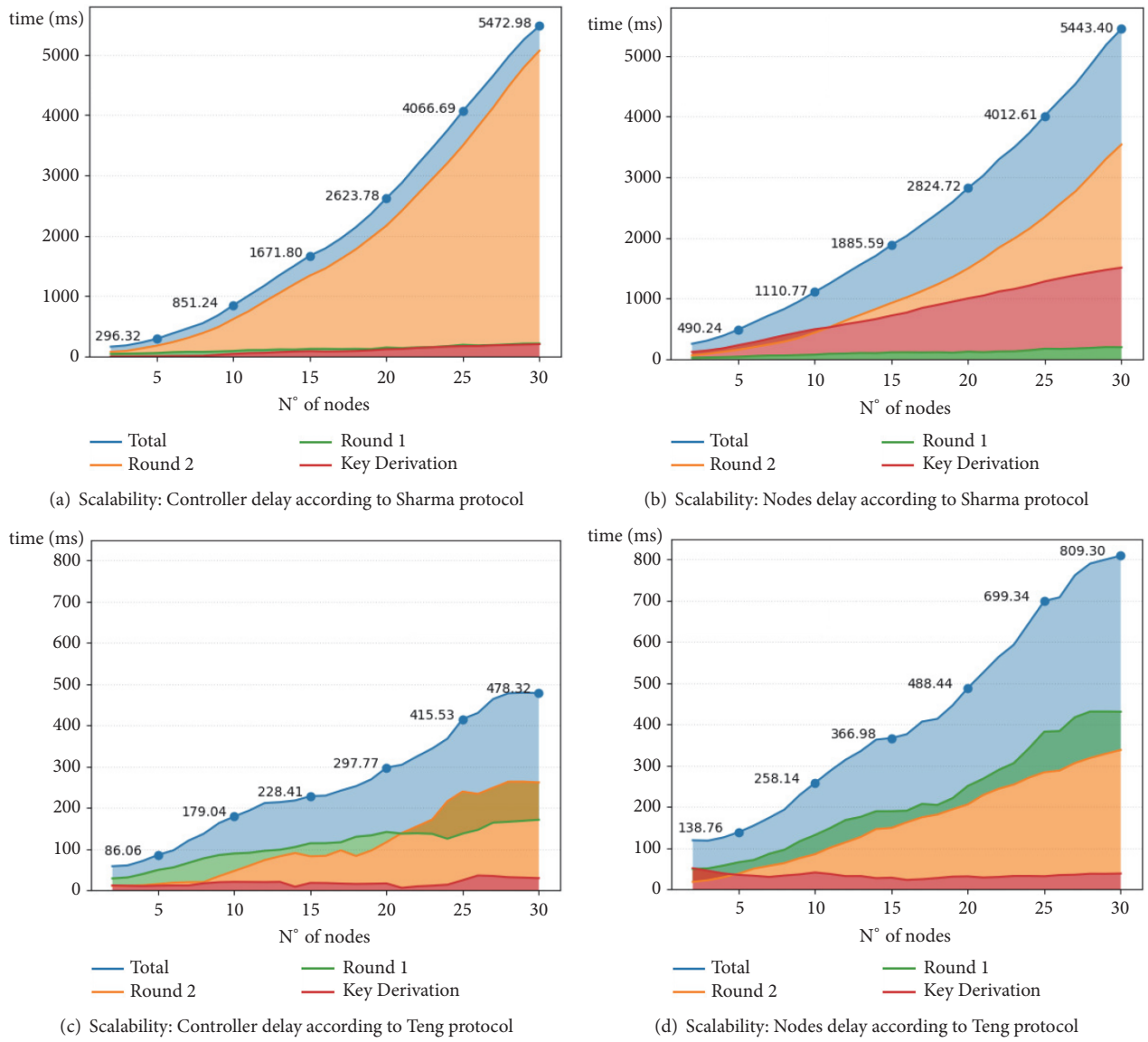
(6) When the controller receives the `ROUND_1_Reply` message from all the participants of the group, it will send `ROUND_2` messages by waiting for `ROUND_2_REPLY` messages.

(7) When the controller receives `ROUND_2_REPLY` messages from all switches, the key computation of group key is started.

(8) As a last step the switches that belong already into a group can start exchanging `DATA` messages with controller by using OpenFlow protocol.

Following the SDN concept, we validate the SSPSoc protocol by using the emulator Mininet 3 [29], running on a computer. As far as the network entities: OpenVSwitches (OVS) [30] was used as SDN switches and a Ryu [31] was used as an SDN controller. The network hosts are emulated using lightweight OS-level virtualisation: each virtual host inside the mininet network corresponds to a container and it has a virtual network interface with a distinct IP address [21]. Applications, such as the PKG, controller, and node executables, can run directly inside virtual hosts. In our experiments, the hosts are interconnected using virtual Ethernet links and OVS switches running in kernel mode. In each emulated network instance, one virtual host was used to run the PKG, one host for the controller, and the rest of the hosts to run the nodes participating in the GKA. As far as the implementation of GKA protocols, we used the Pairing Based Cryptography (PBC) [32] cryptographic library, RIPEMD-160, SHA-256 hash functions, and AES-GCM cipher. Following the PBC library, Type A (based on symmetric pairing) and Type d159 (based on MNT curves [19]) parameters were used for the implementation of [26] and [25], respectively.

5.1. Network Performance. We perform the scenario for a sample of 1 up to 30 nodes (32 virtual hosts in total). Specifically, in order to test the performance of our protocol based on group key agreement, groups of 2 up to 30 nodes (switches) were created. In this work, the first concern was the evaluation of the performance of our SSPSoC by using two different GKA protocols in order to find out which is more appropriate in our case according to their total cost, the cost of `ROUND_1` and `ROUND_2`, and the key derivation cost. The total cost is referring to the time when a first `INVITE` message has been broadcasted until the time that the first `DATA` message has been formed and sent. The cost of two



(a) Scalability: Controller delay according to Sharma protocol

(b) Scalability: Nodes delay according to Sharma protocol

(c) Scalability: Controller delay according to Teng protocol

(d) Scalability: Nodes delay according to Teng protocol

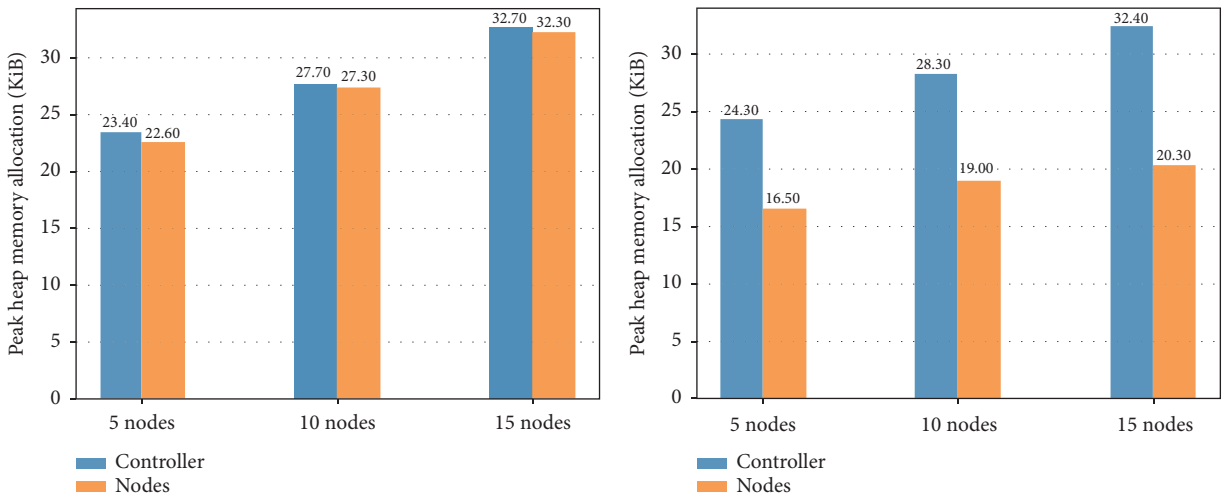
Figure 5: Performance results.

rounds is referring to the period from first ROUND_1 message or ROUND_2 message accordingly sent by controller until the period that the last ROUND_1_REPLY or ROUND_2_REPLY message received by the switch. The key derivation cost refers to time that we need in order to derive the group key (Figure 5).

5.2. Memory Consumption. Following the MPSoC concept another important factor we should take into account is the memory consumption, since both GKA protocols are promising low power consumption. We perform the above scenario for 5, 10, and 15 nodes. The total amount of heap memory allocated during the execution of the SSPSoC protocol by using the two GKA schemes was measured with Valgrind tool Suite [20], which perform a dynamic binary analysis and enables the Massif heap profiler. The results are presented on Figure 6.

6. Conclusion and Future Work

In this paper, we propose a new communication protocol based on group key agreement approach able to address the inside communication of a CoC system. Following the design of the proposed SSPSoC protocol, we validate and simulated it within an SDN environment. Our results focused on the evaluation of two GKA schemes according to their scalability and their power consumption. As far as the results are concerned, we noticed that the Teng protocol has far better performance and significantly lower power consumption based on the number of participants and that makes it more appropriate option for the third phase of our SSPSoC protocol. From the other side the Sharma protocol, even without using pairing as Teng protocol, has higher cost and memory consumption. We believe that these results are obtained due to the authenticity of every participant that



(a) Sharma protocol (b) TENG protocol

Figure 6: Memory consumption of two GKA protocols.

the Sharma protocol is considering in contrast to the Teng protocol which does not consider the authenticity of the group participants.

Possibly, future work could be the implementation of the SSPSoC protocol within the context of MPSoC. In order to achieve this, a cyclic accurate simulator should be chosen and some appropriate modifications should be made in the SSPSoC protocol. Another possible extension could be to add more controllers in our system and create groups of controllers; with this way secure communication is provided not only between ICs but also between the different abstraction layers of a MPSoC architecture.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] P. Sethi and S. R. Sarangi, "Internet of things: architectures, protocols, and applications," *Journal of Electrical and Computer Engineering*, vol. 2017, Article ID 9324035, 25 pages, 2017.

[2] G. Bousdras, F. Quitin, and D. Milojevic, "Template architectures for highly scalable, many-core Heterogeneous SoC: could-of-chips," in *Proceedings of the 13th International Symposium on Reconfigurable Communication-Centric Systems-on-Chip, ReCoSoC 2018*, pp. 1–7, IEEE, July 2018.

[3] S. Kumar, A. Jantsch, J.-P. Soininen et al., "A network on chip architecture and design methodology," in *Proceedings of the IEEE Computer Society Annual Symposium, VLSI, 2002*, pp. 117–124, IEEE, 2002.

[4] P. Ezhumalai, A. Chilambuchelvan, and C. Arun, "Novel NoC topology construction for high-performance communications," *Journal of Computer Networks and Communications*, vol. 2011, Article ID 405697, 6 pages, 2011.

[5] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: past, present, and future of programmable networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[6] L. Cong, W. Wen, and Z. Wang, "A configurable, programmable and software-defined network on chip," in *Proceedings of the 2014 IEEE Workshop on Advanced Research and Technology in Industry Applications, WARTIA 2014*, pp. 813–816, IEEE, September 2014.

[7] R. Sandoval-Arechiga, J. L. Vazquez-Avila, R. Parra-Michel, J. Flores-Troncoso, and S. Ibarra-Delgado, "Shifting the network-on-chip paradigm towards a software defined network architecture," in *Proceedings of the International Conference on Computational Science and Computational Intelligence, CSCI 2015*, pp. 869-870, IEEE, USA, December 2015.

[8] R. Sandoval-Arechiga, R. Parra-Michel, J. L. Vazquez-Avila, J. Flores-Troncoso, and S. Ibarra-Delgado, "Software defined networks-on-chip for multi/many-core systems: a performance evaluation," in *Proceedings of the 12th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, ANCS 2016*, pp. 129-130, USA, March 2016.

[9] A. Scionti, S. Mazumdar, and A. Portero, "Software defined Network-on-Chip for scalable CMPs," in *Proceedings of the 14th International Conference on High Performance Computing and Simulation, HPCS 2016*, pp. 112–115, Austria, July 2016.

[10] K. Berestizshevsky, G. Even, Y. Fais, and J. Ostrometzky, "SDNoC: software defined network on a chip," *Microprocessors and Microsystems*, vol. 50, pp. 138–153, 2017.

[11] S. Ellinidou, G. Sharma, J.-M. Dricot, and O. Markowitch, "A SDN solution for system-on-chip world," in *Proceedings of the 5th International Conference on Software Defined Systems, SDS 2018*, pp. 14–19, Spain, April 2018.

[12] N. McKeown, T. Anderson, H. Balakrishnan et al., "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[13] D. Samociuk, "Secure communication between OpenFlow switches and controllers," in *Proceedings of the AFIN 2015: The Seventh International Conference on Advances in Future Internet*, vol. 39, 2015.

[14] H. Zhang, Z. Cai, Q. Liu, Q. Xiao, Y. Li, and C. F. Cheang, "A survey on security-aware measurement in SDN," *Security and Communication Networks*, vol. 2018, Article ID 2459154, 2018.

[15] G. Sharma, V. Kuchta, R. A. Sahu, S. Ellinidou, O. Markowitch, and J. Dricot, "A Twofold Group Key Agreement Protocol for NoC based MPSoCs," in *Proceedings of the 2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pp. 1-2, August 2018.

[16] G. Sharma, S. Ellinidou, V. Kuchta, R. A. Sahu, O. Markowitch, and J.-M. Dricot, "Secure communication on noc based mpsoc," in *Proceedings of the International Conference on Security and Privacy in Communication Systems*, pp. 417–428, Springer, 2018.

[17] O. N. Foundation. OpenFlow Switch Specification Version 1.5.1 (Protocol version 0x06), 2015.

[18] R. Klöti, V. Kotronis, and P. Smith, "OpenFlow: a security analysis," in *Proceedings of the 2013 21st IEEE International Conference on Network Protocols, ICNP 2013*, pp. 1–6, Goettingen, Germany, October 2013.

[19] A. Miyaji, M. Nakabayashi, and S. Takano, "New explicit conditions of elliptic curve traces for FR-reduction," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E84-A, no. 5, pp. 1234–1243, 2001.

[20] N. Nethercote and J. Seward, "Valgrind: a framework for heavyweight dynamic binary instrumentation," *ACM SIGPLAN Notices*, vol. 42, no. 6, pp. 89–100, 2007.

[21] R. Rong and J. Liu, "Distributed mininet with symbiosis," in *Proceedings of the 2017 IEEE International Conference on Communications, ICC 2017*, pp. 1–6, IEEE, May 2017.

[22] S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "SDN security: A survey," in *Proceedings of the 2013 Workshop on Software Defined Networks for Future Networks and Services, SDN4FNS 2013*, pp. 1–7, IEEE, Trento, Italy, November 2013.

[23] T. Dierks, "The transport layer security (TLS) protocol version 1.2," Tech. Rep. RFC5246, 2008.

[24] K. Benton, L. J. Camp, and C. Small, "OpenFlow vulnerability assessment," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN '13)*, pp. 151-152, ACM, August 2013.

[25] G. Sharma, R. A. Sahu, V. Kuchta, O. Markowitch, and S. Bala, "Authenticated group key agreement protocol without pairing," in *Proceedings of the International Conference on Information and Communications Security*, pp. 606–618, Springer, 2017.

[26] T. Jikai and W. Chuankun, "An identity-based group key agreement protocol for low-power mobile devices," *Chinese Journal of Electronics*, vol. 25, no. 4, pp. 726–733, 2016.

[27] P. Syverson, "A taxonomy of replay attacks [cryptographic protocols]," in *Proceedings of the The Computer Security Foundations Workshop VII*, pp. 187–191, IEEE, Franconia, NH, USA, 1994.

[28] M. J. Dworkin, "Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC," Tech. Rep. SP 800-38D, 2007.

[29] Mininet Project. 2017. http://mininet.org/.

[30] Linux Foundation, "OvS Open vSwitch," 2016. http://www.openvswitch.org.

[31] Ryu SDN Framework Community, "Component-based software defined networking framework build SDN agilely," 2017. https://osrg.github.io/ryu/.

[32] Ben Lynn, PBC Library. 2018. https://crypto.stanford.edu/pbc/.

Hindawi
Security and Communication Networks
Volume 2019, Article ID 8690853, 12 pages
https://doi.org/10.1155/2019/8690853

Research Article

SoftME: A Software-Based Memory Protection Approach for TEE System to Resist Physical Attacks

Meiyu Zhang,[1] **Qianying Zhang**,[1,2] **Shijun Zhao**,[3] **Zhiping Shi**,[1,4] **and Yong Guan**[1,5]

[1] *College of Information Engineering, Capital Normal University, Beijing 100048, China*
[2] *Beijing Advanced Innovation Center for Imaging Theory and Technology, Beijing 100048, China*
[3] *Institute of Software Chinese Academy of Sciences, Beijing 100190, China*
[4] *Beijing Key Laboratory of Light Industrial Robot and Safety Verification, Beijing 100048, China*
[5] *Beijing Key Laboratory of Electronic System Reliability Technology, Beijing 100048, China*

Correspondence should be addressed to Qianying Zhang; qyzhang@cnu.edu.cn

Received 29 November 2018; Accepted 6 February 2019; Published 4 March 2019

Guest Editor: Sascha Uhrig

Copyright © 2019 Meiyu Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The development of the Internet of Things has made embedded devices widely used. Embedded devices are often used to process sensitive data, making them the target of attackers. ARM TrustZone technology is used to protect embedded device data from compromised operating systems and applications. But as the value of the data stored in embedded devices increases, more and more effective physical attacks have emerged. However, TrustZone cannot resist physical attacks. We propose SoftME, an approach that utilizes the on-chip memory space to provide a trusted execution environment for sensitive applications. We protect the confidentiality and integrity of the data stored on the off-chip memory. In addition, we design task scheduling in the encryption process. We implement a prototype system of our approach on the development board supporting TrustZone and evaluate the overhead of our approach. The experimental results show that our approach improves the security of the system, and there is no significant increase in system overhead.

1. Introduction

The development of the Internet of Things (IoT) [1, 2] is hailed as the third wave of world information development after computers and the Internet. Embedded systems play a major role in the information processing of IoT. Embedded systems emerged in the 1970s and have been applied to various fields. They have high requirements for security and real-time [3]. Human beings are exposed to more and more smart devices in our daily lives, most of which are embedded systems. These devices store valuable personal information, making them the target of attackers and resulting in software attacks and memory data leakage accidents from time to time. To enhance the security of embedded systems, ARM proposed the TrustZone [4] security extension to its CPU architecture, which provides resource access control and memory isolation to protect sensitive data. ARM TrustZone technology plays an important role in information protection and has important applications on mobile embedded devices [5–7]. It divides hardware resources into a secure world and a normal world and builds an isolated trusted execution environment (TEE) for applications to protect trusted applications from compromised operating systems and applications [8].

As the value of the data stored in embedded devices increases, effective physical attack methods have emerged. From a cost perspective, inexpensive physical attacks allow attackers to launch attacks more easily, making them more serious practical threats to embedded systems. Some embedded devices are easily lost, such as mobile phones and personal computers, and the working environment of some embedded devices is unsupervised, which makes embedded devices more vulnerable to physical attacks. Such attacks mainly refer to cold boot attacks, bus monitoring attacks, and DMA (direct memory access) attacks. A cold boot attack obtains data by attacking physical memory. A bus attack can obtain data transmitted on the bus, and a DMA attack allows an attacker to use DMA interface of the device to obtain data. Attackers extract and analyze the data in the memory

to make illegal profits, making research on the security of embedded systems critical. Unfortunately, TrustZone does not enforce memory encryption, so it cannot resist above physical attacks [9]. Therefore even if sensitive information is stored in physical memory protected by TrustZone, an attacker can gain valuable information through inexpensive physical attacks. A solution to protect TEE against physical attacks is required.

In response to these issues, we propose SoftME, an approach to protect the confidentiality and integrity of sensitive applications. In our approach, we use TrustZone technology to allocate the on-chip memory space to the secure world, and execute TEE OS on the on-chip memory to protect against cold boot attacks. On-chip memory communicates with the core via the on-chip bus, so it can resist bus attacks. We use data encryption to protect the security of data transmission and storage off-chip. More specifically, the data is encrypted on the on-chip memory before being written back to the off-chip memory. That is to ensure that the data is always in the form of ciphertext when transmitted off-chip. We also design task scheduling for encryption and task execution, enabling parallel execution of tasks on a single processor. Compared to existing solutions, our approach has the following advantages. On the one hand, our design is based on software, without the need for additional hardware support. On the other hand, our approach does not need to modify the applications, and they can be executed directly. We implemented our prototype on the development board supporting TrustZone and evaluated its performance through experiments. The experimental results show that our approach improves the security of embedded systems and there is no significant increase in performance overhead.

In summary, we make the following contributions:

(i) We propose SoftME, an approach that uses the on-chip memory to defend against physical attacks. This approach allocates the on-chip memory space to the secure world of TrustZone, with no additional hardware support and no need to modify applications.

(ii) We protect the confidentiality and integrity of off-chip data by authenticated encryption algorithm and guarantee the fairness of the encryption process and task execution process through task scheduling.

(iii) We implement our prototype system on a physical development board supporting TrustZone, and the experimental results show that our approach improves system security and does not significantly increase system overhead.

The rest of the paper is organized as follows. Section 2 presents related work. Section 3 introduces background knowledge. Section 4 discusses our threat models and assumptions. Section 5 describes our design in detail and the security analysis of our approach. The implementation is described in Section 6, followed by the performance evaluation in Section 7. Finally, Section 8 concludes this paper.

2. Related Work

An attacker can obtain sensitive information in an embedded device through physical attacks. In order to resist physical attacks, the academic field has proposed various security solutions. This section briefly introduces these solutions, which can be divided into hardware assistance and software enhancements.

2.1. Hardware Assistance. As early as 2000, the Stanford University Computer Systems Laboratory implemented an execution only memory in the form of the hardware that supported internal compartments, and the compartments could not access each other [10]. In addition, in [11], the authors proposed a hardware-based memory protection scheme which protected the integrity of a single-core processor and the confidentiality of multiprocessor shared memory. In [12], they described the hardware mechanism of the SecBus project. The project used a separate hardware module to implement encryption protection, designed to protect the memory bus against bus attacks. To achieve the on-chip memory encryption, the Dartmouth College research team designed the Bear microkernel operating system [13, 14]. The Bear operating system provided an encryption mechanism using a security-enhanced processor to process data that appeared outside of the processor. In [15], the authors used a dedicated hardware detector to detect and prevent malicious attacks.

Hardware-assisted protection scheme has an advantage in performance; however, the disadvantages of integrating hardware components are that they occupy the space of embedded devices and increase energy consumption. Our approach relies on the on-chip memory which is available on many embedded development boards, and our approach is based on software to defend against physical attacks.

2.2. Software Enhancements. For embedded systems, sensitive data can be protected by encrypting memory. Hong D et al. proposed three different encryption methods for processing different sensitivity data. DynapoMP [16] was the first method to consider dividing the on-chip memory into different regions. For the problem of excessive memory encryption overhead, Papadopoulos P et al. implemented a secure memory allocator (s_malloc) [17] to allocate any size of memory from the heap dynamically, and any data written to this part of the memory would be encrypted. In addition, in [18, 19], the authors enhanced data security by redesigning the operating system. In [20], they proposed processor-memory bus encryption using the technique of locking cache. Similarly, in [9], Zhang N et al. also created a cache-based independent execution environment by locking the cache. In [21], the researchers proposed SecureME to hide data from compromised operating system and built a secure computing environment for applications. In [22], the authors proposed vTZ, a solution for virtualizing TrustZone. It provided a secure and isolated execution environment between guest TEEs. For physical attacks, Guan et al. proposed Copker [23], an encryption engine implemented inside the CPU, and they

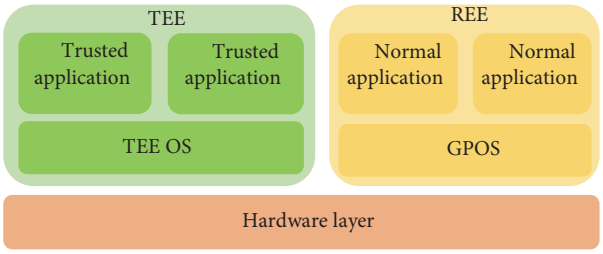


Figure 1: TEE architecture.

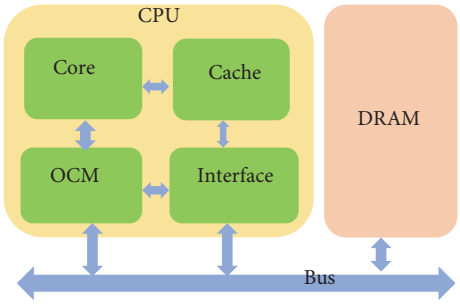


Figure 2: Embedded system based on the on-chip memory.

proved that it provides a secure service against cold boot attacks. In [24], the authors proposed Loop-Amnesia, a disk encryption technology that eliminated cold boot attacks. The mechanism is a kernel-based software design that provides protection for encryption keys in RAM and CPU.

Most of above designs require modifications to the operating systems and cannot be arbitrarily ported to other hardware platforms. One of the advantages of our approach is that we have implemented it in software without any modification to operating systems and it can be used to protect data within an acceptable overhead to the system.

3. Background

In this section, we will introduce the background knowledge related to our work. We will introduce the TEE architecture and ARM TrustZone first, and then we will discuss the on-chip memory architecture of ARM processor.

3.1. TEE and ARM TrustZone. Trusted execution environment (TEE) is proposed by GlobalPlatform. It is a trusted computing environment to provide security services to applications [25, 26]. TEE architecture is shown in Figure 1. The rich execution environment (REE) provides the execution environment for most functional applications, forming a dual operating system architecture with TEE [27]. In the dual operating system architecture, the GPOS and the TEE OS are executed on the same hardware platform, and hardware isolation based technology forms a secure execution environment for TEE OS. GPOS provides general functions to meet the functional requirements of the system, while TEE OS provides protection to trusted applications.

TrustZone is a set of secure hardware extension mechanisms for ARM processors to build an isolated computing execution environment for trusted applications [28]. ARM TrustZone technology reconfigures the processor, using the TrustZone Address Space Controller (TZASC) and TrustZone Protection Controller (TZPC) to divide the hardware resources into two separate parts, called the secure world and the normal world. In order to switch between the two worlds, ARM processor adds a new mode, monitor mode. The software running in this mode is called monitor, which is responsible for saving the context and restoring the state being switched while switching the worlds. In the CP15 coprocessor with the ARM processor, there is a Security Configuration Register (SCR) with an NS bit that indicates the current state of the processor. The secure world and the normal world can be switched by setting the NS bit in the monitor mode. If the NS bit is 1, it means that the current processor state is the normal world, and if it is 0, it means that the current processor state is the secure world. The processor can enter the monitor mode from the normal world by calling SMC (Secure Monitor Call) instruction or hardware interrupts.

3.2. On-Chip Memory Architecture. The application of the cache greatly improves processor performance, but it has unpredictable data access time. The emergence and widespread use of the on-chip memory solves this problem [29]. On-chip memory (OCM) is a general term for static random-access memory (SRAM) that is integrated into the chip for non-cache use. Compared with cache, OCM has the advantages of low power consumption, high performance, and small footprint. Banakar R et al. show that OCM reduces power consumption by 40% and the on-chip area by about 34% compared with the cache of the same capacity [30]. A typical embedded system with OCM is shown in Figure 2. The OCM and the cache communicate with off-chip memory through an off-chip data interface. Unlike off-chip memory, the on-chip memory and core communicate through the on-chip bus, so the on-chip memory is more secure than the off-chip memory.

In addition, OCM has a separate address space, and its address space is often mapped to the memory space. In this case, the entire address space is divided into two parts, OCM only occupies a small part, and the rest is still allocated to the off-chip memory. Therefore, the access from processor to OCM can be directly performed the same as the off-chip memory. Accordingly, the access mode of the on-chip memory greatly improves the speed of data reading and writing without going through the caching process. As a result, more and more embedded processors are beginning to integrate OCM on-chip to improve system performance.

4. Threat Model and Assumptions

To crack embedded systems to obtain valuable information, attackers have designed a variety of attack methods in the embedded system hardware level. In this section, we will

describe the threat model related to our design firstly and then introduce the assumptions of our design.

4.1. Threat Model. In our work, we focus on physical memory leaking attacks, such as cold boot attacks, bus monitoring attacks, and DMA attacks. For physical side-channel attacks, they require specialized protection techniques, and we do not consider this kind of attacks in this paper. We also do not consider physical attacks against the SoC, such as invasive attacks. This kind of attacks is targeted at information inside chips and requires specialized and expensive attack equipment, such as laser cutting system, microprobing station, oscilloscope, and focusing ion beam workstation. Only knowledgeable attackers can perform such attacks. So performing such attacks is quite expensive. The value of this equipment may far exceed the value of the targets being attacked. The attack process is also complicated, and even an experienced attacker may take several months, resulting in higher attack costs [31]. Therefore, attacks such as physical side-channel attacks, code injection attacks, and other complex physical attacks are out of the scope of our research.

4.1.1. Cold Boot Attacks. Cold boot attacks are a new type of physical attacks, which have become a part of many popular security threat models. In a cold boot attack, the attacker utilizes the data remanence effect of the memory to obtain the key and valuable information stored in the off-chip memory [32]. Experiments have shown that after the device is powered off, the data on the DRAM does not disappear immediately but will remain for a while [33]. Measurements of DDR1 and DDR2 demonstrate a correlation between temperature and RAM remanence, indicating that even if the surface temperature of the RAM module is slightly cooled by 10°C, the remanence effect is significantly extended [34]. One way to launch the cold boot attack is that an attacker utilizes this period of time to physically remove the DRAM from the target board and put it into the device that the attacker has prepared in advance to read the content in the DRAM. Another method is to restart the target board into an operating system controlled by the attacker to output memory content. Some research groups have successfully performed cold boot attacks on Android smartphones and retrieved private information such as encryption keys, address books, and photos from RAM [35].

4.1.2. Bus Monitoring Attacks. Buses are the crucial information transmission channels between various functional components, which have become the primary targets of attackers. Typical examples of such attacks are bus snooping attacks and bus tampering attacks. For example, an attacker can mount an FPGA board to the bus of an embedded system. By configuring and controlling the FPGA, the attacker can steal or even modify the data transmitted on the bus, thereby disrupting system execution.

Another typical example for bus monitoring attacks is in [36]. In the original Xbox game system, keys were stored in plaintext and transmitted over the South Bridge bus. The attackers exploited bus snooping and injecting to capture or insert information in the bus between system components to obtain the keys and decrypt the secure bootloader, thereby destroying the system trust chain. Subsequently, the attackers developed a low-cost chip that could be soldered to the game system bus, allowing users to bypass the security monitoring mechanism to play pirated games.

4.1.3. DMA Attacks. By configuring the devices that can use the DMA port, an attacker can bypass software security mechanisms and directly read the physical memory. One solution to such attacks is to utilize the IOMMU [37]. The operating system can program the IOMMU to limit the range of memory that the DMA device can access and even deny the DMA device access to memory. For devices equipped with IOMMU, IOMMU makes it impossible for malicious devices to access memory by DMA attacks. However, IOMMU is not available for every device. Fortunately, most ARM platforms support TrustZone, and DMA requests from compromised OS can be rejected to protect secure memory.

4.2. Assumptions. We assume that the ARM platform supports TrustZone technology and is equipped with the on-chip memory. On-chip memory is trusted to protect against physical attacks such as cold boot attacks, while off-chip memory and peripherals are untrusted. We also assume the existence of a device key in the SoC and it is trusted [38], such as KNOX's Device-Unique Hardware Key (DUHK).

5. System Architecture and Design

In this section, we will introduce the design of SoftME (a Software-based Memory Encryption protection approach) in detail based on the previously discussed techniques and assumptions. Finally, we will make a security analysis of our approach.

5.1. System Architecture. In our approach, we use the system architecture described in Figure 3. TrustZone divides hardware resources into two worlds, a secure world and a normal world. The memory of the secure world consists of two parts, one is the on-chip memory space and the other is a small portion of the DRAM. TEE OS and the monitor run on the on-chip memory, and the secure DRAM is used to store encrypted trusted tasks. Most of the space in DRAM is allocated to the normal world for GPOS. The monitor runs in monitor mode and is responsible for handling hardware interrupts and switching between the two worlds. Due to the resource isolation mechanism of TrustZone, GPOS cannot access devices and resources of the secure world, such as IO peripherals and memory, so TEE OS will not be affected by compromised operating systems and applications. For communication between the two worlds, we allocate a small piece of memory on the off-chip memory as shared memory. We design a task scheduler and a memory protection engine (MPE) in the TEE OS. The task scheduler is responsible for scheduling multiple tasks to ensure fair execution. The memory protection engine is used to decrypt or encrypt tasks while reading data from the on-chip memory or writing data

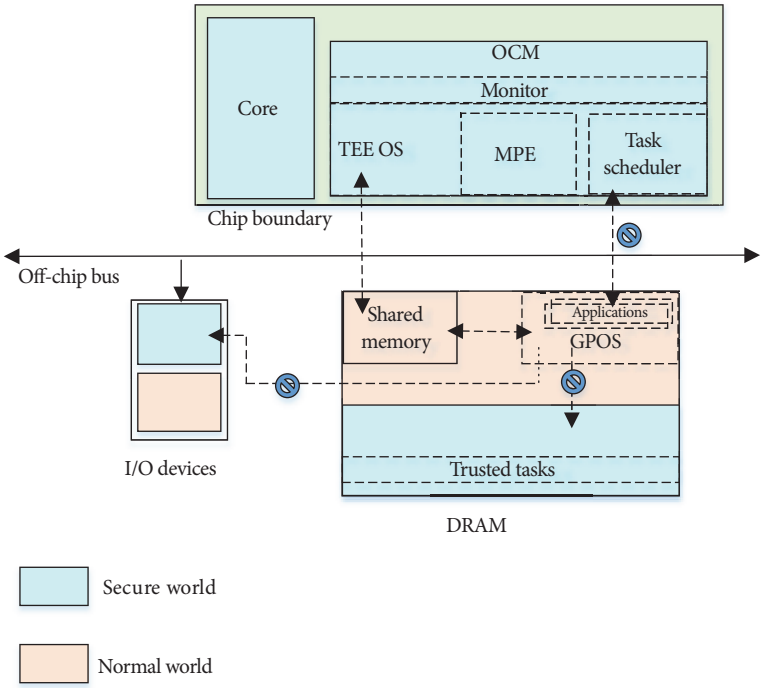


Figure 3: The architecture of SoftME.

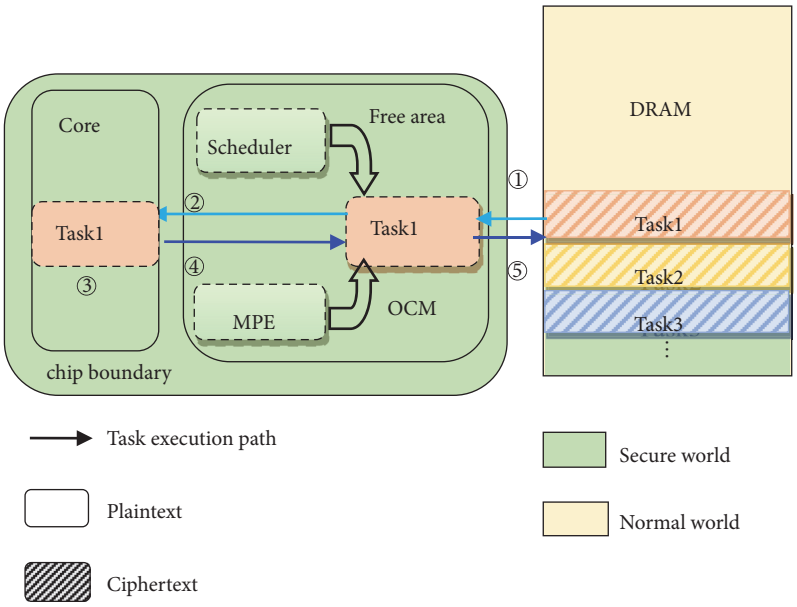


Figure 4: On-chip memory encryption workflow.

to the off-chip memory to ensure data confidentiality and integrity.

5.2. Memory Protection Engine Workflow. In order to prevent sensitive information from being stolen and tampered with, we must ensure the confidentiality and integrity of the data stored on the DRAM. Therefore, we design a memory protection engine in the TEE OS: when a task needs to switch from OCM to DRAM, the memory protection engine encrypts and protects the integrity of the data; when a task needs to be loaded from DRAM to OCM, the memory engine decrypts it and performs an integrity check. The specific process is shown in Figure 4. The execution process of a task in SoftME includes three phases: the loading phase, the

Table 1: Decryption process in the loading phase.

Get the unique key K_t.
Data transfer from DRAM to OCM.
1. Read the T and IV stored on the OCM.
2. $P = AE'(K_t, C, IV, T)$.

Table 2: Encryption process in the switching phase.

Initialization: derive a key from the device key:
K_t = HKDF (K_d, taskID).
Read plaintext from the OCM.
1. $(C, T) = AE(K_t, P, IV)$.
2. $IV = IV + 1$.
3. Store C to DRAM.
4. Store T, K_t and IV to OCM.

activation and execution phase, and the switching phase. Next we will give a detailed design description of these three phases.

(a) The Loading Phase. The loading phase corresponds to step 1 of Figure 4. This phase is to load the task from off-chip memory to the on-chip memory and decrypt it. Before the loading phase, we assume that the task on the off-chip memory has been encrypted. We use the authenticated encryption (AE) algorithm to protect the confidentiality and integrity of the task. The AE algorithm combines the message authentication code and the encryption algorithm to ensure the confidentiality and integrity of the data. Encrypting plaintext can ensure the confidentiality of the data and integrity authentication can verify whether data has been tampered with [39]. The generation of the unique key K_t for each task and the encryption process of the task will be described in the third phase. At this phase, we assume that the key K_t, ciphertext C, initialization vector (IV) IV, and the tag value T are already known. The tag value is the output parameter of the encryption process and is used to verify the integrity of data.

The description of the decryption process is shown in Table 1. The decryption algorithm is represented by the symbol AE', and the encryption algorithm is represented by AE. The ciphertext is the encrypted task read from the off-chip memory. The input parameters of the process are ciphertext C, IV, K_t, and T. If the authentication to the ciphertext is successful, then the plaintext is output; otherwise the symbol *FAIL* is returned. If the authentication fails, it indicates that the ciphertext has been tampered with by an attacker, and the task cannot be recovered. After obtaining the plaintext of the task, we put the task to the ready queue.

(b) The Activation and Execution Phase. The activation and execution phase corresponds to steps 2-4 of Figure 4. In this phase, a task is activated by the operating system and execution begins, just like the normal task execution process. In this phase, the task runs on the on-chip memory. On-chip memory is non-cached SRAM, so it can be read directly by the processor.

(c) The Switching Phase. The switching phase corresponds to step 5 of Figure 4. This phase includes encrypting the data after a task has been completed and storing the ciphertext to DRAM.

Each task has a unique key K_t. As mentioned earlier, we assume that the device key K_d exists and is trusted. We use K_d to derive K_t via the HMAC-based Key Derivation Function (HKDF) [40]. The HKDF algorithm is a key generation function based on the Hash-based Message Authentication Code (HMAC). There are two input parameters for HKDF, a device key and a message. We use the taskID created in the second phase as a message. Since each task has a unique taskID, the key K_t for each task is unique. The device key is highly confidential, and only the kernel has permission to read and operate it. Even if one task obtains the taskID of some other task, it cannot operate the device key, so it cannot obtain other task's key.

The encryption process in this phase is shown in Table 2. The input parameters of the process are the key value K_t, the plaintext P, and the initialization vector IV. The initialization vector is a random number with a fixed length, which is generally 16 bytes. The initialization vector value is incremented by 1 before each encryption. The GCM specification states that IV does not require randomness, but requires that the same key does not use the same IV [41]. We increment the IV by 1 to get different IV values. The plaintext is the task that the memory protection engine reads from the on-chip memory after the task execution is completed. The output values are the ciphertext C and the authentication tag T. After all these steps are completed, the ciphertext will be swapped to DRAM for storage. The tag value, key, and IV of a task will be stored on the on-chip memory.

5.3. Scheduler Workflow. In order to achieve fair scheduling for multitasking, we use the method of time slice polling to design the task scheduler. When a task is created, it will be given a priority, such as high priority, medium priority, or low priority. Tasks in the ready state with the same priority form a ready queue. Take the high-priority ready queue as an example. The workflow of the scheduler is shown in Figure 5. Assume that there are four tasks in the high-priority queue. Currently, task1 is in the running state, and the other three tasks are in the ready state. Suppose our time slice polling time is set to 1ms. During the time period of 0-1ms, task1 is running. At the time of 1ms (assuming that the time of state switching is tiny enough to be ignored), the scheduler sets the state of task1 from the running state to the ready state and places it at the end of the ready queue. Therefore, task2 gets the CPU resources, starts running, and so on. The task scheduler ensures that each task has the same fair execution time. Therefore, even if the CPU is single-threaded, at the high level, it still implements multiple tasks.

5.4. Security Analysis. Sensitive data is valuable to attackers. It is possible to be attacked by an attacker while the task

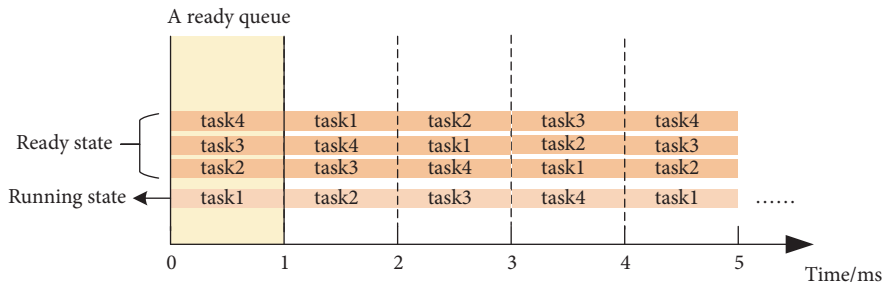


Figure 5: Workflow of the task scheduler.

is running and data is being transferred and stored. In this section, we will prove that SoftME is able to resist the physical attacks listed in the threat model.

Cold boot attacks. Operating systems and applications are vulnerable to cold boot attacks at runtime. In our approach, we run TEE OS and monitor on the on-chip memory. Compared to the off-chip memory, the on-chip memory has the following two features to prevent cold boot attacks. First, the cold boot attack needs to restart the device for subsequent attacks. For the on-chip memory, no matter for how long, after the device is powered off or restarted, the firmware on the board will initialize the on-chip memory and clear all its contents immediately [42], so the attacker could not get the confidential contents on the on-chip memory. In contrast, at room temperature, the off-chip memory can retain a portion of the content (0.1%) even after two seconds when the device is powered off. Second, the cold boot attack is launched when the bootloader of GPOS is started. At that time, the processor state has been the normal world, so the malicious code of the cold boot attack runs in the normal world and it cannot tamper with the on-chip memory, which has been partitioned to the secure world. To sum up, the on-chip memory will not be attacked by cold boot attacks.

Bus monitoring attacks. The on-chip memory is also secure against bus monitoring attacks, because sensitive data never leaves the on-chip memory and it is not transmitted over any exposed off-chip bus. However, the on-chip memory space is limited, it is impossible to store all sensitive data, and a part of sensitive data has to be stored in off-chip memory. For off-chip sensitive task, it can be protected by encryption. SoftME is designed to ensure that data does not appear in plaintext on the off-chip memory or on the bus. The authenticated encryption algorithm also generates a tag value while generating the ciphertext. This value is stored on the chip and used to perform integrity check on the data during decryption.

DMA attacks. On ARM platforms that support TrustZone, DMA attacks are also ineffective. Because DMA reads data directly from DRAM, the data we store on DRAM is processed by encryption. And the on-chip memory is allocated to the secure world, TrustZone will prevent illegal devices from accessing secure world memory through the DMA interface.

6. Implementation

In this section, we will detail the experimental environment and implementation of SoftME.

We implement our design on the Freescale i.MX6q SABRE Lite Board. It features Cortex A9 processor at 1 GHz per core, 1 GB of 64-bit wide DDR3, and 256K on-chip memory. The bootloader for hardware initialization and system boot is provided by onboard flash. The trusted operating system used for the prototype system is version 1.4.0 of TOPPERS/FMP, the general-purpose operating system kernel is Linux 3.10.53, and the monitor for world switching is SafeG 1.2.4. The serial port UART3 is selected as the information transmission port.

6.1. Memory Isolation. We build the SoftME architecture described in Figure 3 on the development board. Figure 6 shows the memory address arrangement of the prototype system. The memory system of the platform consists of two parts, on-chip memory and off-chip memory (DRAM). On-chip memory is all assigned to secure world, running the lightweight trusted embedded operating system TOPPERS/FMP. DRAM is divided into two parts, one for the secure world to store trusted applications and the other for normal world to run Linux. SafeG [43] is used as the monitor for world switching and runs on the on-chip memory. Since FMP, SafeG, and Linux storage space cannot be overlapped, we will store them separately in the memory area, where 0x00900000-0x0091FFFF is used by FMP, and 0x00920000-0x0092FFFF is used by SafeG. 0x12000000-0x4EFFFFFF of the off-chip memory space is used by Linux. The memory protection engine and task scheduler are running in FMP. The remaining size of the on-chip memory is calculated by the total size of the on-chip memory minus the size occupied by the TEE OS and the monitor. In our experimental platform, the total size of the on-chip memory is 256K, the size allocated for FMP kernel (including the memory protection engine and task scheduler) is 128K, and the size allocated for the monitor SafeG is 64K, so the remaining free space is 64K. Therefore, the maximum size of trusted tasks can be about 64K.

6.2. Port TEE OS to the On-Chip Memory. According to the above analysis, the off-chip memory is insecure and

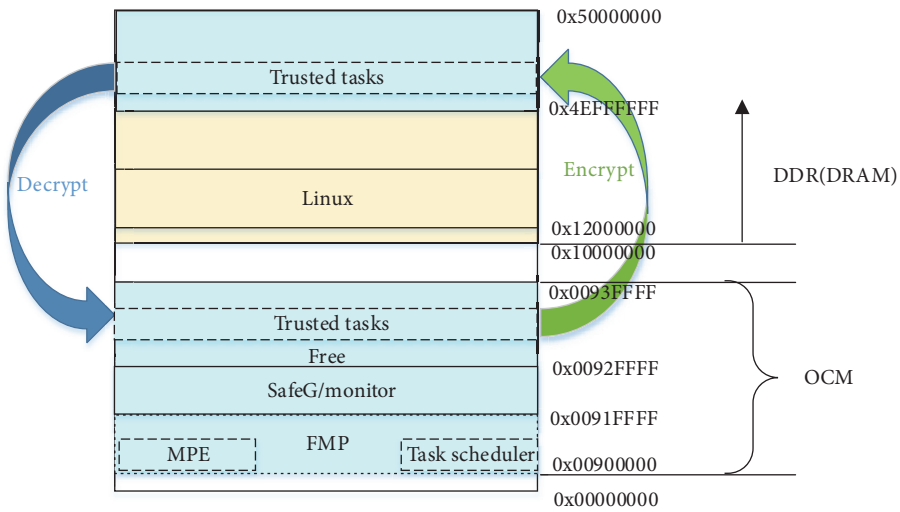


Figure 6: Memory address arrangement.

vulnerable to physical attacks. An important step in implementing SoftME is that the TEE OS should be executed on the on-chip memory. According to the i.MX 6Quad processors reference manual, the physical memory address range of the on-chip memory is 0x00900000-0x0093FFFF (Figure 6), and the entire on-chip memory region can be used freely after booting. We modified the FMP target-dependent configurable parameter value *TARGET_T_OS_START_ADDRESS* to be 0x00907000, which is in the on-chip memory address space range. In addition, the text base address and data base address of FMP are also modified to the physical address range of the on-chip memory. Therefore, after the system boots, FMP will be executed on the on-chip memory.

6.3. The Implementation of the Task Scheduler and the Memory Protection Engine. TOPPERS/FMP is a new generation trusted operating system kernel developed by Japanese TOPPERS project team that follows the μITRON 4.0 specification. μITRON is a real-time multitasking system specification that has become the Japanese industry standard. In terms of creating and activating tasks, we use APIs that comply with μITRON standards. The code sample for creating and activating the task scheduler is described in Table 3. We use the system static function, *cre_cyc ( )*, to create a task scheduler in the configuration file. An ID will be given to the task when the task is created. *sta_cyc ( )* is used for activating a task scheduler and the task state management function *irot_rdq ( )* is used for rotating task precedence. The task scheduler is activated in the TEE OS application to be available.

The role of the memory protection engine is to process the sensitive data on the DRAM. After executing on the on-chip memory, the task data is encrypted and then written back to the DRAM. The encryption key is derived from the device key. The code sample for creating and activating the memory protection engine is described in Table 4. In terms of implementation, we still use the functions of μITRON to create and activate tasks. *cre_tsk ( )* is used to statically create tasks in the configuration file, and *act_tsk ( )* activates tasks. The encryption algorithm we use is AES_GCM_128. We build a cryptography library by using Galois/Counter Mode (GCM). GCM is a kind of authenticated encryption algorithm with counter mode and message authentication code. Its performance evaluation has shown excellent performance on many platforms [44]. The value of parameter *mode* is either *encrypt* or *decrypt*. *encrypt* indicates that the process is an encryption process, and *decrypt* indicates that the process is a decryption process.

Table 3: Code sample of the task scheduler.

```
/* Create a task scheduler */
cre_cyc (taskid);
/* Implementation */
void task_scheduler (void)
{
irot_rdq (TASK_PRIORITY);
}
/* Activate the task scheduler */
sta_cyc (taskid);
```

Table 4: Code sample of the memory protection engine.

```
/* Create a memory protection engine */
cre_tsk (taskid);
/* Memory protection engine implementation */
void engine (void)
{
......
mbedtls_gcm_crypt_and_tag (mode, K_t, IV, P/C);
......
}
/* Activate the memory protection engine */
act_tsk (taskid);
```

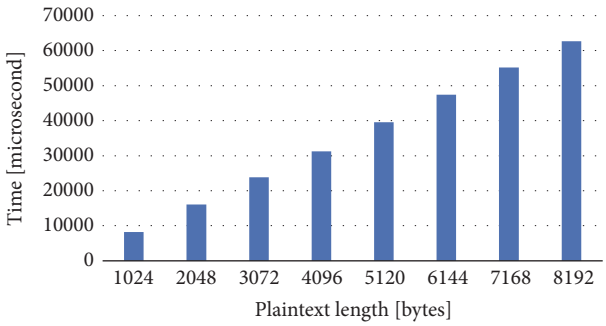


Figure 7: The overhead of the memory protection engine.

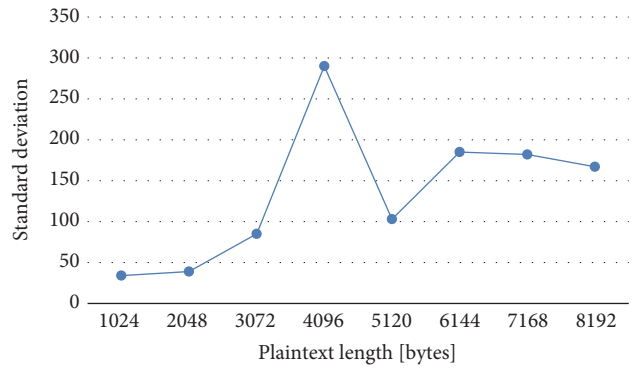


Figure 8: Standard deviation reflecting the degree of data dispersion.

Owing to the tiny size of the cryptography library we built, we can easily integrate the memory protection engine into FMP and port them to the on-chip memory. However, μITRON has a standard real-time kernel specification for any small embedded system design, including FMP, which results in some problems in the process of importing cryptographic algorithms in our experiment. More specifically, FMP developers use a custom function library specified by the standard; therefore if we directly imported the cryptographic algorithm to an FMP application, library conflict will be thrown. To address this problem, we replace the library file imported by memory protection engine with the FMP library file.

7. Experimental Results and Analysis

In this section, we evaluate the prototype system and give an analysis of the experimental results. In the experiment, we measured the overhead of the memory protection engine by encrypting different size of plaintext in the prototype system. Then we designed several trusted tasks for evaluation and analyzed the impact of the memory protection on the overhead of these tasks.

7.1. Code Modification. To coordinate the execution of the system, we need to patch the Linux kernel to support the execution of the monitor. The specific modifications are summarized as follows. We modified a total of 26 lines of code for 5 files in the bootloader, mainly some macro definitions related to switching the two worlds and the name of some startup file. A total of 2696 lines of code for 7 files are modified in Linux, mainly some conditional statements related to switching the worlds and SMC call to SafeG. Finally, we modified a total of 263 lines of code for two communication related configuration files in Linux to communicate with SafeG.

7.2. Overhead of Memory Protection Engine. First of all, we measured the overhead of the memory protection engine in the prototype system, with encrypting data whose length ranges from 1K to 8K. As demonstrated in Figure 7, the overhead of the memory protection engine increases almost linearly with the increase of the plaintext size.

Standard deviation is the most commonly used form of quantization that reflects the degree of dispersion of a set of data. Figure 8 shows the standard deviation of the data we collected. It can be seen that the overhead is variable and not very stable.

7.3. Overhead on Multitasks. In this section, we show the overhead of our approach in terms of the task execution time, the task switching time, and the task preemption time, and we analyze the experimental results separately.

To observe the impact of encryption on the simultaneous execution of multiple tasks, we designed six same tasks for evaluation. Since the time of each time slice of the FMP can be set to a minimum of 1 millisecond, in order to make the experimental results obvious, the execution time of each task we designed is longer than 1 millisecond.

For multiple tasks, we enable the tasks to be executed simultaneously by activating the task scheduler. To observe the impact of the memory protection engine on multitasks, we design two encryption strategies for the memory protection engine. The first strategy is that all the tasks are encrypted by only one memory protection engine. In this case, all tasks data cannot be encrypted at the same time and may generate waiting time. In the second strategy, multiple tasks are managed by multiple memory protection engines. That is, when the system creates a task, it also creates a memory protection engine accordingly. Owing to the tiny size of the memory protection engine, it will not have a significant impact on the system performance. In the experiment, each case is compared with the corresponding basic case tasks. These basic tasks are executed on the same experiment platform without the memory protection.

For the above three cases, we executed 1, 2, 4, and 6 tasks and measured their overhead separately. We have run 50 times for each case, and Figure 9 shows the experimental results and the comparison of the three cases. The overhead of a task execution includes task execution time, task switching time, and dispatch time of the task scheduler. It can be seen from the figure that the running time is almost linearly proportional to the number of tasks. Table 5 illustrates the overhead introduced by data encryption and shows the proportional relationship between strategy1/strategy2 and the basic case, respectively. For the first encryption strategy, the overhead of a task execution is increased by about 50%

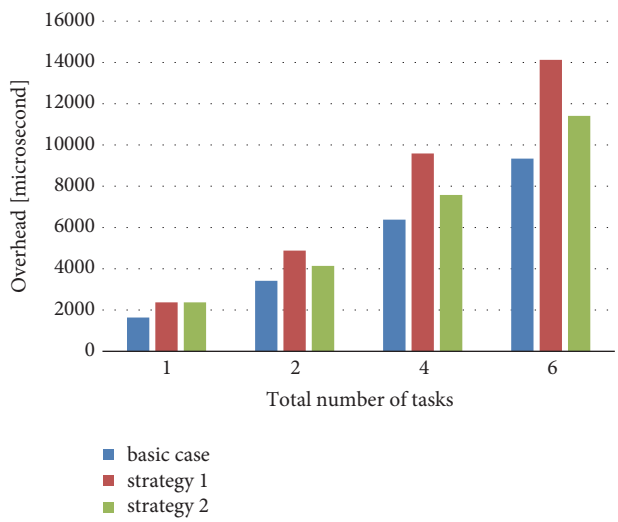


Figure 9: The overhead of different protection strategies.

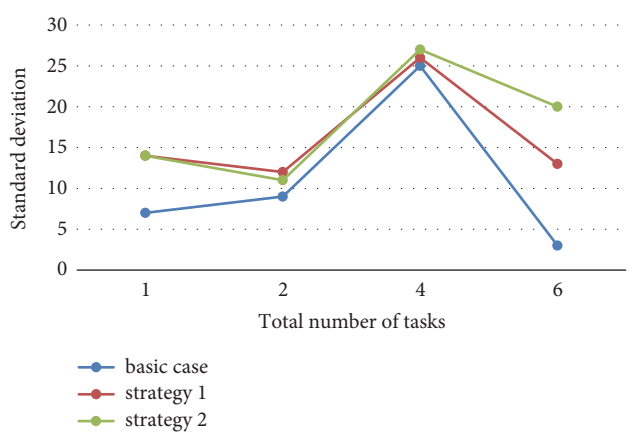


Figure 10: Standard deviation reflecting the degree of data dispersion.

compared to the basic case. When multiple tasks are executed, the advantages of the second strategy are obvious, since the overhead increases only about 20%.

Figure 10 lists the standard deviation of the overhead. Overall, these three cases are more stable than the overhead of the memory protection engine shown in Figure 8. We think it is because the execution time of the task hides the overhead of the memory protection engine. Among the three cases, the basic case is relatively stable, and we can see that although the second strategy has an advantage in terms of overhead, its stability is slightly worse than the first strategy.

As can be seen from above experimental results, in the embedded real-time operating system, ensuring the security of the tasks will inevitably affect the real-time task. Therefore, for a task requiring high real-time performance, we keep it on the on-chip memory. Therefore, the task does not need to be decrypted when it runs again. We measured the task preemption time and the task switching time for multitasking on the on-chip memory. The task preemption time is the time required for a high-priority task to preempt a low priority

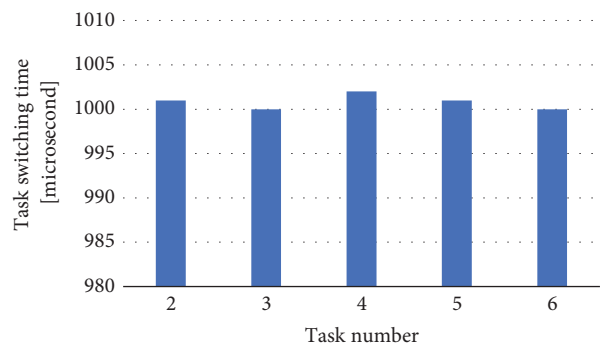


Figure 11: The task switching time.

Table 5: The proportional relationship between the two strategies and the basic case.

Number of tasks\ strategies	Strategy 1	Strategy 2
1	1.45	1.45
2	1.43	1.21
4	1.50	1.19
6	1.51	1.22

Table 6: The task preemption time.

	Average time/μs
Linux	8.56
basic case	0.86
strategy1	0.87
strategy2	0.86

task. It includes the time to save the context of the preempted task and the time to resume the context of the high-priority task. We measured the task preemption time in the Linux kernel, the FMP kernel without the memory protection, the FMP kernel running strategy1, and the FMP kernel running strategy2, respectively. The evaluation tasks are two tasks with different priorities, task1 and task2. The priority of task1 is set high, and the priority of task2 is set low. Task1 is activated in task2, so task1 can preempt task2. Using the system timer, the time before activating task1 and the time before running the first instruction of task1 are recorded, respectively. We run the experiments 100 times, and the average times are shown in Table 6. It shows that the FMP kernel has an absolute advantage over Linux in processing real-time tasks, and it can be seen that SoftME has no effect on the task preemption time.

The task switching time is the time spent on switching one task to another task. We activated six tasks at the same time and recorded the time when the first instruction of each task was invoked, denoted by t1 to t6. So the time spent on switching from task1 to task2 is t2-t1, the time spent on switching from task2 to task3 is t3-t2, and so on. The experimental result is shown in Figure 11. In Section 5.3, we configured the task switching time to be 1ms. It can be seen from the figure that the actual time is more than 1ms, which

is due to the need to save and restore the context when the tasks are switched.

8. Conclusions and Future Work

IoT devices have the disadvantage of being unsupervised, making them vulnerable to physical attacks. We propose SoftME, an approach for protecting trusted tasks against physical attacks. We use the on-chip memory to protect TEE OS and design a memory protection engine to protect the confidentiality and integrity of the tasks stored on the off-chip memory. We implemented SoftME on the physical development board. Finally, the experimental evaluation shows that the memory protection introduces an overhead of about 20%, which is within acceptable limits.

However, for a single-core embedded system, encryption will have a negative impact on the execution of real-time tasks. Multicore architecture has replaced single-core systems in many areas and has become the mainstream of embedded systems. In a multicore architecture, multiple tasks can be executed in parallel, with more parallel computing power, lower clock frequency, lower power consumption, and higher efficiency. Therefore, in future work, we can allocate a dedicated core for the memory protection engine, and other cores are responsible for executing tasks. This design will make use of the parallel computation of multicore to reduce the overhead caused by encryption and decryption, thereby improving the performance.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research was supported by the National Key R & D Plan (2017YFB1301100), the National Natural Science Foundation of China (61602325, 61802375, 61702348), the Project of Beijing Municipal Education Commission (KM20190028005), the Project of the Beijing Municipal Science & Technology Commission (LJ201607), and Capital Normal University Major (key) Nurturing Project.

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): a vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] F. Wortmann and K. Flüchter, "Internet of things: technology and value added," *Business & Information Systems Engineering*, vol. 57, no. 3, pp. 221–224, 2015.

[3] P. Koopman, "Embedded system security," *Computer*, vol. 37, no. 7, pp. 95–97, 2004.

[4] P. Varanasi and G. Heiser, "Hardware-supported virtualization on ARM," in *Proceedings of the Second Asia-Pacific Workshop on Systems (APSys '11)*, ACM, July 2011.

[5] N. Santos, H. Raj, S. Saroiu et al., "Using ARM trustzone to build a trusted language runtime for mobile applications," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 1, pp. 67–80, 2014.

[6] J. Y. Hwang, S. B. Suh, S. K. Heo et al., "Xen on ARM: system virtualization using xen hypervisor for ARM-based secure mobile phones," in *Proceedings of the 5th IEEE Consumer Communications and Networking Conference (CCNC'08)*, pp. 257–261, IEEE, Las Vegas, Nev, USA, January 2008.

[7] S. Yoo, Y. Liu, C. H. Hong, C. Yoo, and Y. Zhang, "Mobivmm: a virtual machine monitor for mobile phones," in *Proceedings of the 1st ACM Workshop on Virtualization in Mobile Computing*, pp. 1–5, ACM, June 2008.

[8] M. Sabt, M. Achemlal, and A. Bouabdallah, "Trusted execution environment: what it is, and what it is not," in *Proceedings of the 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2015*, pp. 57–64, IEEE, Finland, August 2015.

[9] N. Zhang, K. Sun, W. Lou et al., "Case: cache-assisted secure execution on ARM processors," in *Proceedings of the 2016 IEEE Symposium on Security and Privacy, SP 2016*, pp. 72–90, IEEE, USA, May 2016.

[10] D. Lie, C. Thekkath, M. Mitchell et al., "Architectural support for copy and tamper resistant software," *ACM SIGPLAN Notices*, vol. 35, no. 11, pp. 168–177, 2000.

[11] W. Shi, H. H. S. Lee, M. Ghosh et al., "Architectural support for high speed protection of memory integrity and confidentiality in multiprocessor systems," in *Proceedings of the 13th International Conference on Parallel Architectures and Compilation Techniques*, pp. 123–134, IEEE Computer Society, 2004.

[12] L. Su, A. Martinez, P. Guillemin et al., "Hardware mechanism and performance evaluation of hierarchical page-based memory bus protection," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*, p. 180, 2009.

[13] M. Henson and S. Taylor, "Beyond full disk encryption: protection on security-enhanced commodity processors," in *Proceedings of the International Conference on Applied Cryptography and Network Security*, pp. 307–321, Springer, Berlin, Germany, 2013.

[14] M. Henson and S. Taylor, "Attack mitigation through memory encryption of security enhanced commodity processors," in *Proceedings of the 8th International Conference on Information Warfare and Security (ICIW'13)*, D. Hart, Ed., pp. 265–268, USA, March 2013.

[15] D. Arora, S. Ravi, A. Raghunathan, and N. K. Jha, "Hardware-assisted run-time monitoring for secure program execution on embedded processors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 12, pp. 1295–1308, 2006.

[16] D. Hong, L. A. D. Bathen, S.-S. Lim, and N. Dutt, "DynaPoMP: dynamic policy-driven memory protection for SPM-based embedded systems," in *Proceedings of the 6th Workshop on Embedded Systems Security*, ACM, Taiwan, 2011.

[17] P. Papadopoulos, G. Vasiliadis, G. Christou, E. Markatos, and S. Ioannidis, "No sugar but all the taste! memory encryption without architectural support," in *European Symposium on Research in Computer Security*, vol. 10493 of *Lecture Notes in*

Computer Science, pp. 362–380, Springer, Cham, Switzerland, 2017.

[18] L. Guan, P. Liu, X. Xing et al., "TrustShadow: secure execution of unmodified applications with ARM trustzone," in *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services*, pp. 488–501, ACM, USA, June 2017.

[19] W. Huang, V. Rudchenko, H. Shuang, Z. Huang, and D. Lie, "Pearl-TEE: supporting untrusted applications in trustzone," in *Proceedings of the 3rd Workshop on System Software for Trusted Execution*, pp. 8–13, ACM, Toronto, Canada, October 2018.

[20] X. Chen, R. P. Dick, and A. Choudhary, "Operating system controlled processor-memory bus encryption," in *Proceedings of the Design, Automation and Test in Europe, DATE'08*, pp. 1154–1159, IEEE, 2008.

[21] S. Chhabra, B. Rogers, Y. Solihin et al., "SecureME: a hardware-software approach to full system security," in *Proceedings of the International Conference on Supercomputing*, pp. 108–119, ACM, 2011.

[22] Z. Hua, J. Gu, Y. Xia et al., "vTZ: virtualizing ARM trustzone," in *Proceedings of the 26th USENIX Security Symposium*, 2017.

[23] L. Guan, J. Lin, Z. Ma et al., "Copker: a cryptographic engine against cold-boot attacks," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 742–754, 2018.

[24] P. Simmons, "Security through amnesia: a software-based solution to the cold boot attack on disk encryption," in *Proceedings of the 27th Annual Computer Security Applications Conference*, pp. 73–82, ACM, 2011.

[25] S. Pinto, T. Gomes, J. Pereira et al., "IIoTEED: an enhanced, trusted execution environment for industrial iot edge devices," *IEEE Internet Computing*, vol. 21, no. 1, pp. 40–47, 2017.

[26] Y. Fan, S. Liu, G. Tan et al., "One secure access scheme based on trusted execution environment," in *Proceedings of the 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (Trustcom/BigDataSE)*, pp. 16–21, IEEE, USA, 2018.

[27] J. Jang, S. Kong, M. Kim et al., "SeCReT: secure channel between rich execution environment and trusted execution environment," in *Proceedings of the Network and Distributed System Security Symposium*, San Diego, Calif, USA, 2015.

[28] D. Sangorrin, S. Honda, and H. Takada, "Dual operating system architecture for real-time embedded systems," in *Proceedings of the 6th International Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT)*, pp. 6–15, Brussels, Belgium, 2010.

[29] R. Tabish, R. Mancuso, S. Wasly et al., "A real-time scratchpad-centric OS for multi-core embedded systems," in *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS 2016*, pp. 1–11, IEEE, Austria, 2016.

[30] R. Banakar, S. Steinke, B. S. Lee et al., "Scratchpad memory: a design alternative for cache on-chip memory in embedded systems," in *Proceedings of the Tenth International Symposium on Hardware/Software Codesign, CODES 2002*, pp. 73–78, IEEE, USA, 2002.

[31] A. Tria and H. Choukri, "Invasive attacks," in *Encyclopedia of Cryptography and Security*, pp. 623–629, Springer, Boston, Mass, USA, 2011.

[32] S. F. Yitbarek, M. T. Aga, R. Das et al., "Cold boot attacks are still hot: Security analysis of memory scramblers in modern processors," in *Proceedings of the 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 313–324, IEEE, 2017.

[33] M. Gruhn and T. Muller, "On the practicability of cold boot attacks," in *Proceedings of the 2013 8th International Conference on Availability, Reliability and Security (ARES)*, pp. 390–397, IEEE, USA, September 2013.

[34] J. A. Halderman, S. D. Schoen, N. Heninger et al., "Lest we remember: cold-boot attacks on encryption keys," *Communications of the ACM*, vol. 52, no. 5, pp. 91–98, 2009.

[35] M. Tilo, S. Michael, and F. C. Freiling, "Frost: forensic recovery of scrambled telephones," in *Proceedings of the International Conference on Applied Cryptography and Network Security*, pp. 373–388, Banff, AB, Canada, 2013.

[36] A. Huang, "Keeping secrets in hardware: the microsoft Xbox™ case study," in *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems*, vol. 2523 of *Lecture Notes in Computer Science*, pp. 213–227, Springer, Berlin, Germany, 2002.

[37] P. Stewin and I. Bystrov, "Understanding DMA malware," in *Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, vol. 7591 of *Lecture Notes in Computer Science*, pp. 21–41, Springer, Berlin, Germany, 2012.

[38] S. Zhao, Q. Zhang, G. Hu, Y. Qin, and D. Feng, "Providing root of trust for ARM TrustZone using on-chip SRAM," in *Proceedings of the 4th International Workshop on Trustworthy Embedded Devices*, pp. 25–36, ACM, Scottsdale, Ariz, USA, November 2014.

[39] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, T. Okamoto, Ed., vol. 1976 of *Lecture Notes in Computer Science*, pp. 531–545, Springer, Berlin, Germany, 2000.

[40] H. Krawczyk and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)," RFC Editor RFC5869, 2010.

[41] D. McGrew and J. Viega, "The Galois/counter mode of operation (GCM)," Submission to NIST Modes of Operation Process, 2004.

[42] P. Colp, J. Zhang, J. Gleeson et al., "Protecting data on smartphones and tablets from memory attacks," *ACM SIGARCH Computer Architecture News*, vol. 43, no. 1, pp. 177–189, 2015.

[43] D. Sangorrin, S. Honda, and H. Takada, "Integrated scheduling for a reliable Dual-OS monitor," *Information and Media Technologies*, vol. 7, no. 2, pp. 627–638, 2012.

[44] D. A. McGrew and J. Viega, "The security and performance of the Galois/counter mode (GCM) of operation," in *Proceedings of the International Conference on Cryptology in India*, vol. 3348 of *Lecture Notes in Computer Science*, pp. 343–355, Springer, Berlin, Germany, 2004.

Hindawi
Security and Communication Networks
Volume 2019, Article ID 8936784, 11 pages
https://doi.org/10.1155/2019/8936784

Research Article

Multidevice False Data Injection Attack Models of ADS-B Multilateration Systems

Fute Shang, Buhong Wang, Fuhu Yan, and Tengyao Li

School of Information and Navigation, Air Force Engineering University, Xi'an, Shanxi, China

Correspondence should be addressed to Buhong Wang; wbhyl@aliyun.com

Received 11 September 2018; Revised 12 December 2018; Accepted 14 January 2019; Published 3 March 2019

Academic Editor: Stefan Katzenbeisser

Copyright © 2019 Fute Shang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Location verification is a promising approach among various ADS-B security mechanisms, which can monitor announced positions in ADS-B messages with estimated positions. Based on common assumption that the attacker is equipped with only a single device, this mechanism can estimate the position state through analysis of time measurements of messages using multilateration algorithm. In this paper, we propose the formal model of multidevice false data injection attacks in the ATC system against the location verification. Assuming that attackers equipped with multiple devices can manipulate the ADS-B messages in distributed receivers without any mutual interference, such attacker can efficiently construct attack vectors to change the results of multilateration. The feasibility of a multidevice false data injection attack is demonstrated experimentally. Compared with previous multidevice attacks, the multidevice false data injection attacks can offer lower cost and more covert attacks. The simulation results show that the proposed attack can reduce the attackers' cost by half and achieve better time synchronization to bypass the existing anomaly detection. Finally, we discuss the real-world constraints that limit their effectiveness and the countermeasures of these attacks.

1. Introduction

With the rapid development of air transportation, the surveillance technology of air traffic control is gradually evolving from primary surveillance radar technology to higher precision and lower cost Automatic Dependent Surveillance-Broadcast (ADS-B) [1] technology. ADS-B provides surveillance data to support mission-critical automatic and human decisions in the next generation ATC system to increase safety of air traffic. The aircraft equipped with ADS-B transmitter can broadcast plain text of the aircraft position retrieved from their onboard GPS receiver. The messages then are received by ground or airborne ADS-B receiver and displayed on the screens of controllers and pilots [2]. However, due to the fact that ADS-B technology did not consider the security problem comprehensively [3, 4] at the design stage, it is vulnerable to multiple wireless threats [5]. The false data injection attack can be injected into the monitoring system by low cost software defined radio equipment, which can interfere with the work of the controllers and even cause safety problems.

In order to improve protection against the ADS-B false data injection attack, a variety of location verification schemes have been proposed. At present, multilateration technology [6, 7] has been widely used [8]. The advantage of this technology is that it can reduce the cost of deployment by using the existing secondary surveillance radar and ADS-B systems. Given the precise distance between transmitter and four receivers, it is easy to calculate the position of the transmitter, that is, the position of the aircraft [9]. Finally, by comparing the location of the aircraft and the position information in the ADS-B message, it is possible to verify whether the ADS-B message is legitimate and effectively improves the capability of detecting the false data injection attack.

Existing multilateration techniques can effectively detect attackers using a single transmitter, but did not consider the multidevice attack scenario [10]. In this scenario, the attacker uses multiple transmitters distributed in different geographic locations to transmit a ADS-B message that is constructed ingeniously under a specific time delay, controlling the time of receivers to receive the signal, making the location multilateration calculated the same as the location of the attacker's injection, thus bypassing the multilateration verification and injection of large number of false ADS-B data. We

call this new attack pattern multidevice false data injection attack.

Moser et al. [11] first proposed the multidevice false data injection attack of ADS-B multilateration system in ATC surveillance. However, Moser et al. simulate the arrival time of the real transmitter without utilizing the characteristics of the multilateration algorithm. The attacker needs to use the identical number of transmitters and receivers to make the attack success, which increases the cost of an attack. Secondly, as the transmitters are deployed in a large area, it is difficult to realize time synchronization, which makes it possible to be detected by other detection methods based on frequency and phase. Therefore, we consider how to find a better attack scheme, using less attacker transmitters to realize the attack and make the distance between the transmitters within a certain range to effectively realize the time synchronization between the transmitters.

In this paper, we propose a more accurate model of multidevice attacker for multilateration algorithm which utilizes less transmitters and centralized deployment of the antennas to achieve the attack and overcome the shortcomings mentioned earlier. Finally, we propose possible countermeasures for this kind of attack. The main contributions of this article are as follows:

(1) We establish the formal model of ADS-B multilateration system and multidevice false data injection attack.

(2) We proposed the best attack scheme calculation method.

(3) We demonstrate the feasibility of multidevice false data injection attack method.

In Section 2 we introduce the ADS-B system security and multilateration technology. In Section 3 we describe the ADS-B multilateration system and the multidevice false data injection attacker model and find out the requirements of the false data injection attack. In Section 4 we give the implementation of the model and the example validation. Possible countermeasures for this kind of attack are discussed in Section 5 and we conclude in Section 6.

2. Background and Related Work

At present, researchers have proposed a variety of false data injection attacks against ADS-B systems, but most of these attacks can be detected by ADS-B location verification based on multilateration.

2.1. ADS-B Attack Patterns. The ADS-B transmitters emit the ADS-B messages in the signal format specified by the ADS-B protocol standard DO-260 [12], DO-260A [13], and DO-260B [14], and these messages are not encrypted. The characteristics of wireless communication allow attackers easily interference, eavesdrop, and modify, inject, and delete ADS-B messages [15].

2.1.1. Ghost Aircraft Inject. Based on the false data injection attack, the attacker is able to broadcast the ADS-B message generated by the ghost aircraft that does not exist in the surveillance system. The target may be any legitimate ADS-B receiver. When the visibility is low, the ghost aircraft may be confused with legitimate aircraft and the ground air traffic controller may force the aircraft to land or change the track. In the same way, for aircraft targets, the attacker can also make the ADS-B based collision avoidance system generate error instructions. In the case of low visibility, the pilot may perform dangerous operations based on these instructions.

2.1.2. Aircraft Flood Denial. Based on the above ghost aircraft injection attack, the aircraft flood denial attack can inject multiple aircraft simultaneously. The purpose of the attack is to enable the air traffic controller's surveillance system to deny of service. Due to the fact that the air traffic controllers confused real aircraft with ghost aircraft, the surveillance system is defeated.

2.1.3. Virtual Trajectory Modification. The purpose of this attack is to modify the trajectory of real aircraft. First delete all location reports of the target aircraft, and inject modified messages. This attack takes advantage of the low precision of other surveillance technologies, such as the first radar. When the central processing station (CPS) fuses the radar data and ADS-B data, a certain error is allowed. If the distance between the real aircraft position and injected position is very small, the attack will be able to bypass the detection technology and lead to the wrong judgment or response to the collision avoidance system.

2.1.4. False Alarm Inject. Similar to the virtual trajectory modification attack, attackers delete the messages generated by real aircraft and inject false ADS-B messages for alarm. ADS-B provides emergency (such as hijacking) alarm mechanism. False alarm injection attack will lead to an emergency response mechanism, such as a refusal of landing requests. Due to the fact that voice recognition is not reliable in hijacking events, it is difficult to detect such attacks at the physical level.

If an attacker uses a single transmitter to attack, all of the attacks above will result in multilateration results that are not consistent with the ADS-B report location, which will be detected easily.

2.2. Existing Countermeasures. According to the review article by Strohmeier et al. [16], ADS-B protection technology is mainly divided into two categories: secure broadcast authentication [17] and secure location verification [18] technology.

2.2.1. Secure Location Verification Technology. It can identify abnormal ADS-B messages after the attack has launched and then filter them out. Various characteristics of wireless signal are extracted to locate the position of the aircraft, such as receiving signal strength, angle of arrival, frequency of arrival, time of arrival (TOA), and time difference of arrival (TDOA). The location estimation method using angle of arrival is called triangulation, and the one using time of arrival is called multilateration [19].

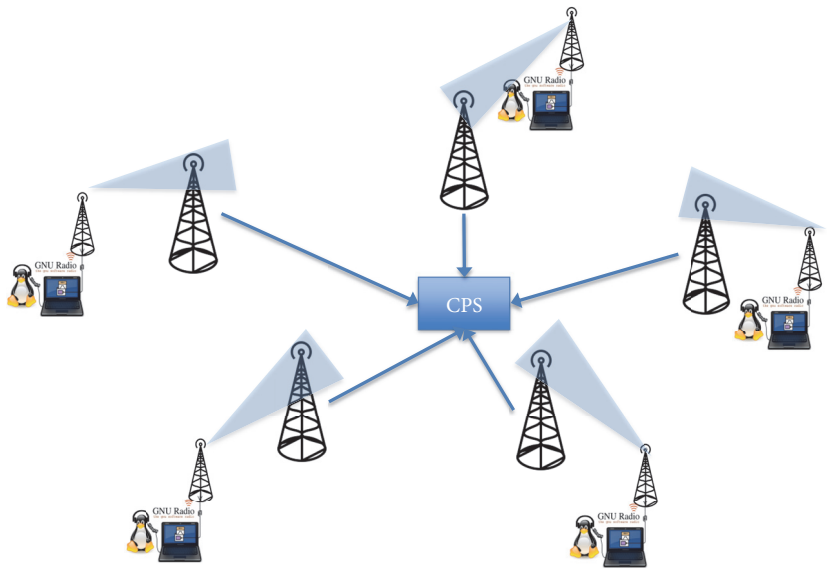


Figure 1: Single-to-single multidevice false data injection attack model.

The advantage of multilateration technology is that it can reuse existing ADS-B devices without any modification to airborne equipment. Multilateration can easily detect the false data injection attack and even locate the position of the attacker transmitter. The central processing station utilizes time difference of arrival of the same messages from different ADS-B receivers to calculate the location of the ADS-B transmitter and compares it with the position information in the ADS-B message. The legitimate ADS-B message with allowed position error will be passed to the Air Traffic Control Center [20] to manage the air traffic.

At present, multilateration technology is widely used in ground location verification. It is used by the ASDE-X system [21] at airports and the Wide Area Multilateration [22, 23]. Although multilateration technology has been widely applied, there are still many problems [24]. For example, it is difficult to estimate altitude information accurately based on ground based receiver. Besides, multilateration technology also needs many receivers to receive the same signal accurately, and all receivers are connected with the central processing station safely.

And we should notice that crowdsourcing is a well-established paradigm in the commercial air traffic tracking domain [25]. Volunteers around the world set up and operate large numbers of receivers for transponder signals and send the live tracking data to a central server via the Internet. This approach has been used to improve the security in GPS [26].

2.2.2. Secure Broadcast Authentication Technology. It is to prevent attacks through the authentication mechanism, which falls into two categories, the noncryptographic schemes on the physical layer and cryptographic schemes. Here we focus on physical layer based technology, which is to find differences between legitimate and nonlegitimate packets based on characteristics of the wireless channel which are hard to replicate.

There are various approaches of physical layer based technology, such as approaches based on received signal strength (RSS, [27, 28]), based on frequency offset caused by imperfections in the transmitters synchronization [11], or based on the carrier phase (e.g., [29]). While these methods can effectively defeat jamming and modification attacks, the inherently lower performance makes them difficult to use in a large-scale system such as ADS-B.

2.3. Multidevice Attack. The above security countermeasures are based on the attacker model of single transmitter, but ignore the scenario of multiple transmitters. With the development of software defined radio equipment, the cost of attackers using multiple transmitters is getting lower and lower. Therefore, the multilateration system with outdated attack models may be unsafe, and a new attack model is needed to study its security [30].

In the multidevice attack proposed by Moser et al., a low-power, small covering radius transmitter is set near each receiver to transmit the signal to ensure that the signal is only injected into the nearest receiver, as shown in Figure 1. Under this configuration, the attacker transmitter can easily control the time delay between transmissions, making the arrival time at each receiver identical to the arrival time generated from the ghost aircraft. As a consequence, the multilateration calculated location is consistent with the position encoded in the malicious ADS-B message, and the attacker bypasses the location verification successfully. However, there are still some shortcomings in this type of attack:

(1) The transponder needs to be near the receiver, once found easy to destroy.

(2) This attacker needs to have the same number of transponders as ADS-B receivers. SDR platforms have different transient phase feature, which is easy to be detect. While high-end arbitrary waveform generators might be able to produce highly accurate signals with little noise in the transient phase, they are by a factor of 20 to 100 times more expensive than today's SDR platforms and would require a prohibitively high budget on the attacker's side since a distributed attack requires at least a couple of such devices.

(3) In practice, it is difficult to synchronize the local oscillators of different devices that are separated over large distance, which make it easy to be detected by frequency based detection methods.

Based on these features, Moser et al. have built ATC intrusion detection system to detect this kind of attack. Therefore, we consider how to find a better attack scheme, using less attacker transmitters to realize the attack and make the distance between the transmitters within a certain range to effectively realize the time synchronization between the transmitters.

3. Multidevice Attack Model of ADS-B Multilateration System

Next, we will establish the mathematical model of the ADS-B multilateration system, find out the requirements to implement the false data injection attack, and finally establish the model of the multidevice false data injection attack.

3.1. ADS-B Multilateration System Model

Definition 1 (ADS-B multilateration system). The ADS-B multilateration system can be defined as a tuple:

$$\mathbf{S} = \left\langle \mathbf{P}^S, \mathbf{X}, \mathbf{Z}^g, \mathbf{Z}^m \right\rangle \tag{1}$$

Suppose we use K ADS-B receivers for multilateration, and the coordinates of these receivers are represented by the $K \times 3$ matrix $\mathbf{P}^S = \{s_{i,j}\}$.

$$\mathbf{P}^S = \begin{bmatrix} s_{11} & s_{12} & s_{13} \\ s_{21} & s_{22} & s_{23} \\ \vdots & \vdots & \vdots \\ s_{K1} & s_{K2} & s_{K3} \end{bmatrix} \tag{2}$$

P_k^S is the position of kth receiver:

$$\mathbf{P}_k^S = \begin{bmatrix} s_{k1} & s_{k2} & s_{k3} \end{bmatrix} \tag{3}$$

$\mathbf{X}$ is the real position of aircraft i, $\mathbf{Z}^g = \{\mathbf{z}_{ij}^g\}$ is the position of aircraft i decoded from the ADS-B message by the receiver j, and $\mathbf{Z}^m = \{\mathbf{z}_{ij}^m\}$ is the time when receiver j received the ADS-B message from aircraft i.

ADS-B multilateration system can estimate transmitter location through multiple receiver's TDOA of the same ADS-B message. Therefore, the receivers need precise time synchronization. The TDOA based multilateration algorithm generally assumes that the error is determined by the Gauss distribution. Given the TDOA of receivers, the least square algorithm is applied to find the optimal estimation of the position of the aircraft.

Definition 2 (location estimation using multilateration algorithm). Given the TOA of certain ADS-B messages $\mathbf{Z}^m$, the corresponding receiver location $\mathbf{P}^S$, and the speed of light v, we can estimate the location of target $\mathbf{x} = (x, y, z)$:

$$\mathsf{est}_\varepsilon\left(\mathbf{z}^m, \mathbf{x}\right) \coloneqq \exists \hat{\mathbf{x}} : \bigwedge_{j \in S} \mathbf{z}_j^m = h_j\left(\hat{\mathbf{x}}\right) \wedge \left\|\hat{\mathbf{x}} - \mathbf{x}\right\| < \varepsilon \tag{4}$$

The propagation time $h_i(\mathbf{x})$ from location $\mathbf{x}$ to receiver $\mathbf{P}_j^S$ is given as

$$h_j\left(\mathbf{x}\right) = \frac{\left\|\mathbf{x} - \mathbf{P}_j^S\right\|}{v} \tag{5}$$

The estimation error of target location has upper bound ε, which consists of the time synchronization error and error generated during signal propagation. If the error does not always satisfy the requirement, it is considered that multilateration algorithm can not converge and the location fails.

Definition 3 (bad data detection). The central processing station receives aircraft location encoded in a group of ADS-B messages with the same content and estimates the aircraft location $\mathbf{x}$ using TDOA of these messages. The residue is the difference between them, which is used to monitor the status of the system. If the residue is smaller than the threshold τ, these messages are valid messages; otherwise, they will be filtered out.

$$\mathsf{mon}_\tau\left(\mathbf{z}^g, \mathbf{x}\right) \coloneqq \left\|\mathbf{z}^g - \mathbf{x}\right\| < \tau \tag{6}$$

There are different methods to select the threshold τ, such as χ^2 test.

Although these attackers have the ability to inject arbitrary messages, they must follow some rules to make the attack success in the real world. The measurements have to satisfy some constraints to ensure the receivers can accept the messages successfully. We can encode the communication domain specific knowledge in the logic formulas:

$$\begin{aligned} &\mathsf{con}\left(\mathbf{z}^m, \mathbf{z}^g\right) \\ &\quad \coloneqq \left(0 < \mathbf{z}^m < \frac{R}{v}, \mathbf{z}^g \in V, abs\left(\mathbf{z}_i^m - \mathbf{z}_j^m\right) > l_m\right) \end{aligned} \tag{7}$$

where R is maximum propagation distance of ADS-B messages and V is a reasonable area where the aircraft could appear. According to the RTCA DO-260, the time length of the message block with preamble is $l_m = 120\mu sec$. To ensure two messages received by one receiver will not overlap and interfere with each other, the difference of TOAs must be larger than that.

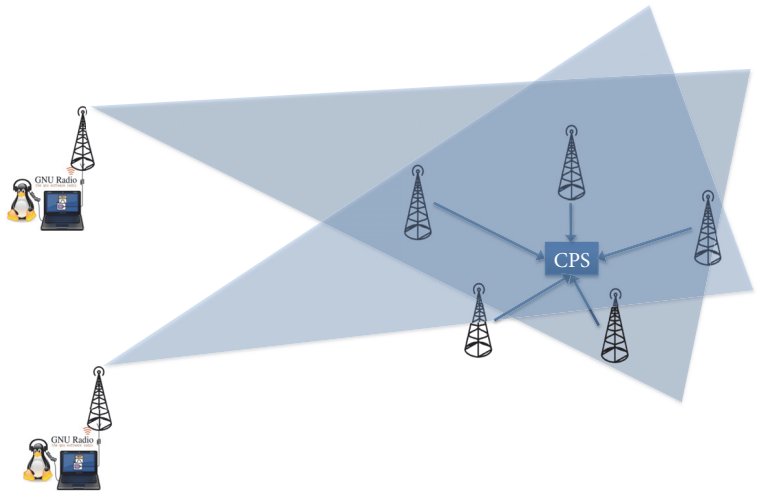


Figure 2: Full connected multidevice false data injection attack model.

3.2. False Data Injection Attack Requirement. Because of the lack of authentication mechanism in ADS-B, attackers can inject false ADS-B messages. Using multilateration mechanism described above, these messages can be labelled as bad data and filter out of the system. In order to realize a false data injection in the ADS-B multilateration system, the following requirements must be satisfied.

Definition 4 (requirement for false data injection attack). Let $\mathbf{S}$ be an ADS-B multilateration system and $\mathsf{est}_\varepsilon(\mathbf{z}^m, \mathbf{x})$ and $\mathsf{mon}_\tau(\mathbf{z}^g, \mathbf{x})$ be as specified above. The requirements of false data inject are defined as follows.

$$\begin{aligned}&\mathsf{fdi}_{\varepsilon,\tau}\left(\mathbf{z}^g, \mathbf{z}^m\right)\\&\quad := \exists \mathbf{x} : \left(\mathsf{est}_\varepsilon\left(\mathbf{z}^m, \mathbf{x}\right) \wedge \mathsf{mon}_\tau\left(\mathbf{z}^g, \mathbf{x}\right) \wedge \mathsf{con}\left(\mathbf{z}^g, \mathbf{z}^m\right)\right)\end{aligned} \tag{8}$$

$\mathbf{z}^g$ are the location claimed by attacker in the ADS-B messages and $\mathbf{z}^m$ are the admissible measurement vector constructed by attacker to satisfy the requirements above to bypass the state estimation and monitor process [31].

In order to meet the requirements of the false data injection attack, a formal model of multidevice false data injection attack is proposed. We analyze the requirements of this new attack model and describe the method of implementing this attack.

3.3. Multidevice False Data Injection Attacker Model. In the formal model of multidevice false data injection attack, we use high-power transmitters to implement message injection attacks at a relatively long distance from the receiver, as shown in Figure 2. To simplify the problem, the multidevice attack model proposed in this paper is based on the following assumptions:

(1) Assume that each receiver can receive all transmitter signals. We consider the communication graph is complete. If the transmitter power is large enough, all receivers will be able to receive the message effectively.

(2) Assume that the receiver only takes the receiving time of the first ADS-B message as the arrival time of the actual signal. In practice, the receiver may receive multiple signals because of the multipath effect of signal propagation. If multiple signals are received, the earliest arrival signal is most likely to be the original signal.

If multilateration systems consider receiving multiple ADS-B messages of the same content at the same station as being attacked, attackers may adopt the combination of message deleting attack and message injection attack to remove the additional messages. As the TOA of the duplicated messages can be predicted in advance, the attacker is able to send an interference signal at the right time to cancel these messages.

Definition 5 (multidevice false data injection attack). Let $\mathbf{S}$ be an ADS-B multilateration system, an attacker wants to inject the ADS-B message of single aircraft at position $\mathbf{z}^g$. A attacker model of multidevice false data injection attack is tuple $\mathbf{A} = \langle \mathbf{P}^A, \Delta t\rangle$, which satisfies

$$\mathsf{fdi}_{\varepsilon,\tau}\left(\mathbf{z}^g, \mathbf{z}^m\right) \tag{9}$$

$\mathbf{P}^A = \{\mathbf{P}_i^A, i = 1..N\}$ is the matrix of attacker's transmitters locations and N is the number of the attacker's transmitters. $\Delta t = \{\Delta t_i, i = 1..N\}$ is the time delay of ADS-B messages transmitted by the attacker, assuming $\Delta t_i = 0$. Redefine the measurement $\mathbf{z}_j^m$ of receiver j as follows, which is the time of arrival used in multilateration estimation $\mathsf{est}_\varepsilon(\mathbf{z}^m, \mathbf{x})$.

$$\mathbf{z}_j^m = \min_{i\in N}\left(h_j\left(\mathbf{P}_i^A\right) + \Delta t_i\right) \tag{10}$$

$h_j(\mathbf{P}_i^A)$ is given in formula (10). As we assume that the receiver only takes the receiving time of the first ADS-B message as the arrival time of the actual signal, the receiver j chooses the minimum of arrival time of all transmitters as its measurement. The arrival time of a transmitter includes

the signal propagation time from transmitter i to receiver j and the time delay Δt_i of transmitter i.

The attacker model is used to describe the transmitters' location and time delay $\langle \mathbf{P}^A, \Delta t \rangle$ to control the arrival time $\mathbf{z}_j^m$ of ADS-B message, which should satisfy the requirement $\mathsf{fdi}_{\varepsilon,\tau}(\mathbf{z}^g, \mathbf{z}^m)$ defined in Definition 4.

4. Implementation of Multidevice False Data Injection Attack

To implement the multidevice FDI attack, we need to find an optimized configuration of the transmitters' location and the transmission delay of each message. According to the configuration, the injected ADS-B message can bypass the multilateration scheme to influence the ATC center. The core of the problem is how to build an optimization model to calculate the transmitters' location and the transmission delay of each message, so that it satisfies the false data injection attack requirement.

4.1. Optimization Problem Establishment. According to Definition 5, we can establish an optimization model of multidevice false data injection attack for ADS-B multilateration system. Let $\mathbf{S}$ be an ADS-B multilateration system; $\mathbf{z}^g$ is the location that attacker want to inject. $\mathbf{P}_i^A$ is the location of attacker transmitter i and Δt_i is the corresponding time delay, which makes the aircraft location estimated by the multilateration the closest to the desired fake position. The problem can be expressed as the following optimization problem:

$$\min \quad \sum_{i=1}^{M} \sum_{\substack{j=1 \\ j \neq i}}^{M} \left(\left\| \mathbf{z}_i^m - \mathbf{z}_j^m \right\| - \left(h_i(\mathbf{z}^g) - h_j(\mathbf{z}^g) \right) \right)^2 \tag{11}$$

$$\text{s.t.} \quad \left\| \mathbf{P}_i^A - \mathbf{P}_j^A \right\| \leq r, \quad i = 1, \ldots, N, \ j = 1, \ldots, N \tag{12a}$$

$$\left\| \mathbf{P}_i^A - \mathbf{P}_j^S \right\| \leq R, \quad i = 1, \ldots, N, \ j = 1, \ldots, M \tag{12b}$$

$$0 < \mathbf{z}^m < \frac{R}{v}, \ \mathbf{z}^g \in V, \ abs\left(\mathbf{z}_i^m - \mathbf{z}_j^m \right) > l_m. \tag{12c}$$

where r is the upper bound of the distance between any two transmitters to meet the time synchronization requirement between the transmitters; R indicates the maximum distance between the arbitrary transmitter and the receiver, which is the ADS-B coverage radius. $\|\mathbf{z}_i^m - \mathbf{z}_j^m\|$ is the measured time difference of arrival (TDOA). $h_i(\mathbf{z}^g) - h_j(\mathbf{z}^g)$ shows the TDOA of the signal emitted from the ghost aircraft. If the difference between two TDOA is the least, the position error is the least.

4.2. Solving the Optimization Problem. The optimization problem described in formula (11) is to solve the multidevice false data injection attack scheme. Then we use the optimized result to evaluate the feasibility and attack efficiency of the attack scheme. To simplify the problem, we use two transmitters to achieve the attack as an example.

In order to simplify the optimization problem, the locations of attacker transmitters are selected at the discrete point of two-dimensional space. First choose the location $\mathbf{P}_1^A$ of the first transmitter in a circular area of radius R and the circular center is the geometric mean of all receiver coordinates so that the ADS-B messages can be normally received between the receivers and the transmitters. The area is discretized and the points in the area are enumerated as the location of the first transmitter. Then, the location of the second transmitter location $\mathbf{P}_2^A$ is enumerated among the discrete points in the second circular area of radius r, the center of which is the first transmitter location $\mathbf{P}_1^A$ to make any two transmitters meet the time synchronization requirements. After determining the position of the two transmitters, the unconstrained single variable optimization problem is constructed to find the best time delay, which makes the position error minimized between the coordinates position calculated by the multilateration and the attacker desired position $\mathbf{z}^g$, as is shown in Algorithm 1.

After solving the optimal attack scheme, we verify the attack configuration. First, we implemented the multilateration algorithm, which can estimate the location of the aircraft by the arrival time of the same message received by 5 receivers. Then, given two transmitter locations, the estimated location changes at different time delays Δt_2 are calculated, as shown in Figure 3. The blue curve indicates that the x coordinate change with the delay Δt_2, the orange curve represents the coordinate y change, and the green curve represents the coordinate z change.

It is found from the graph that, with the linear variation of the transmission delay, the locations estimated by the multilateration will change sharply, which gives us a great convenience to inject continuous flight track.

4.3. Simulation Experiments. Given the number of the receivers, we can get a configuration of the multilateration system by generating the receivers positions randomly. These positions are uniformly chosen in a selected area. Given a certain number of receivers, if the position error in the best solution is larger than 500 meters, we should increase the necessary number of attacker's transmitters. Then we can get the necessary number of transmitters in the configuration after simulation as shown in Table 1. As the increasing of number of receivers, the number of transmitters can be reduced by two. Only under the four-receiver configuration, the attacker can achieve high attack efficiency. Under the condition of 6 receivers, the error is reasonable with 4 receivers. However, it is much bigger than other conditions. It is related to the placement of the receivers. We randomly choose the positions of receivers, which makes the error uncertain. The error is acceptable for this level disturbance, so we do not need extra transmitters for this status.

In order to make the attack more practical, we use real-world data from the OpenSky Network. First, we extract ground stations' locations and select 4 stations from them, which are all in coverage of one ADS-B source and have a dense usage.

This is a configuration of the attackers as shown in Figure 4. If transponders send the message at time offset of $(0.00226, 5.63 \times 10^{-6})$, the multilateration algorithm will

input: ADS-B multilateration system $\mathbf{S}$, the location that attacker want to inject $\mathbf{z}^g$, the upper bound of the distance between any two transmitters r, the maximum distance between the arbitrary transmitter and the receiver R.
output: The location of attacker transmitters $\mathbf{P}^A$, the time delay of the second transmitter Δt.

$\mathbf{P}_c \longleftarrow (1/K)\sum_i \mathbf{P}_i^S$;
$\epsilon \longleftarrow \infty$;
for $r_1 \longleftarrow 1$ **to** R **do**
  for $\theta_1 \longleftarrow 0$ **to** $2\cdot\pi$ **do**
    $\mathbf{P}_1^A \longleftarrow \mathbf{P}_c + (r_1\cdot\cos\theta_1, r_1\cdot\sin\theta_1)$;
    for $r_2 \longleftarrow 1$ **to** r **do**
      for $\theta_2 \longleftarrow 0$ **to** $2\cdot\pi$ **do**
        $\mathbf{P}_2^A \longleftarrow \mathbf{P}_1^A + (r_2\cdot\cos\theta_2, r_2\cdot\sin\theta_2)$;
        $\mathbf{P}^A \longleftarrow (\mathbf{P}_1^A, \mathbf{P}_2^A)$;
        $\Delta\hat{t}, \hat{\epsilon} \longleftarrow \min_{\Delta t}\sum_{i=1}^{M}\sum_{\substack{j=1\\ j\neq i}}^{M}(\|\mathbf{z}_i^m - \mathbf{z}_j^m\| - (h_i(\mathbf{z}^g) - h_j(\mathbf{z}^g)))^2$;
        if $\hat{\epsilon} < \epsilon \wedge \mathrm{con}(\mathbf{z}^m, \mathbf{z}^g)$ **then**
          $\epsilon \longleftarrow \hat{\epsilon}$;
          $\Delta t \longleftarrow \Delta\hat{t}$;
        end
      end
    end
  end
end
return $\mathbf{P}^A, \Delta t$;

Algorithm 1: Implementation of the optimization algorithm (discrete enumeration of the locations of transmitters).

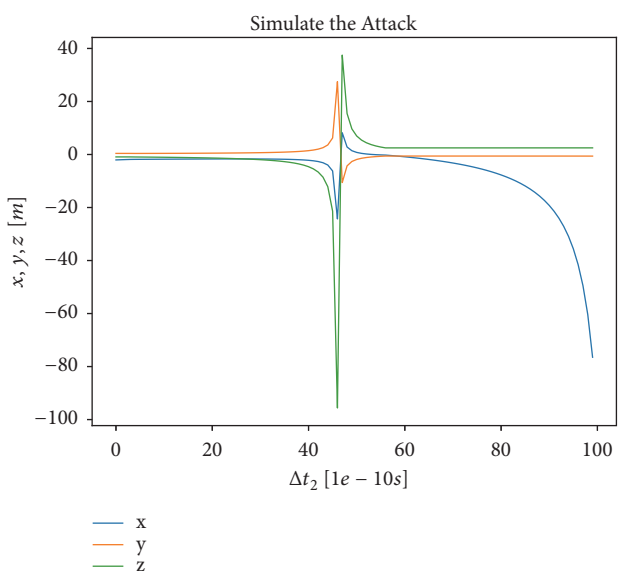


Figure 3: Given the position of attacker's transponder position, the multilateration result variation along with the time delay. The blue curve indicates that the coordinate x change, the orange curve represents the coordinate y change, and the green curve represents the coordinate z change.

estimate the aircraft location at green circle, which is the target of the false message injection. Just ranging the time offset, we can get a trajectory.

To demonstrate the constraints on the trajectory injection, we randomly chose some placements of transmitters and find out what trajectories we can get by changing the time delay between two transmitters. As shown in Figure 5, the trajectories distribute on straight lines, which can be used to inject to the system to interfere with the job of controllers. Figure 5 includes three subfigures. They are chosen from 100 simulations to demonstrate the typical patterns of injected trajectories. To get one subfigure, we first randomly generate the placement of receivers and attackers transmitters. Then we set the time delays of the first transmitter as a sequence of zero, and set the time delays of the second transmitter as an arithmetic sequence start from zero. Given different time delays for different ADS-B messages, the multilateration algorithm can calculate a sequence of estimated targets as shown in the subfigure. However, if the transmitters' location is given, only one possible trajectory could be injected, which is different from the normal trajectory of the aircraft and may be detected by some advanced abnormity detection mechanism. Therefore, attacker model of moving transmitters needs

Table 1: #Receivers = number of receivers in the ADS-B multilateration system, #Transmitters = number of attacker's transmitters in FDI, and Error = the difference between the target location of ghost aircraft and the estimated location calculated by the multilateration.

#Receivers	#Transmitters	Error $[m]$
4	2	44.3417
5	3	45.6391
6	4	107.9898
7	5	67.2437

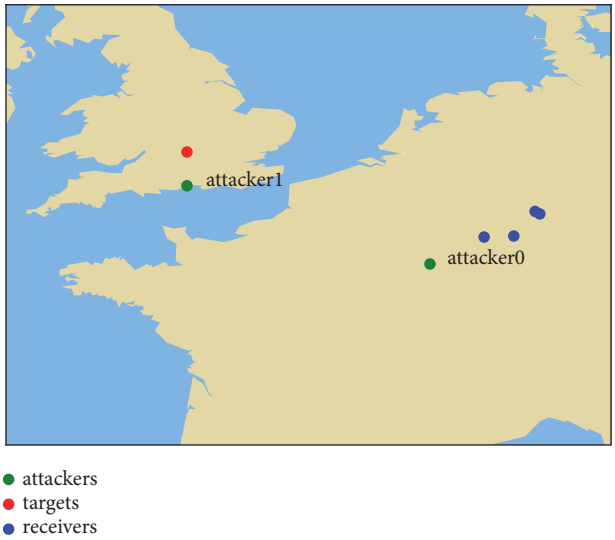


Figure 4: The positions of receivers and attackers. The blue circles are the selected locations of receivers, and the red circles are the locations of attacker's transponders. The green circles are the estimated location of aircraft.

to be studied to construct more realistic trajectories, which has more degrees of freedom.

5. Multidevice False Data Injection Countermeasures

Because the attacker forged the arrival time of the signal, the verification mechanism of the ADS-B multilateration system failed. It is difficult for the existing security mechanisms to detect such attacks effectively. Therefore, we should strengthen the ADS-B multilateration system security. New countermeasures could achieve the goal by hiding the information of locations and operational status.

5.1. Changing Locations. We try to move some of the receivers along a trajectory (by loading them on vehicles) in this countermeasure. They report ADS-B data to the center process station by wireless communication. The multilateration system could get the accurate positions using GNSS on the receive sites. The attackers only have their original information and could not inject false messages at the right time.

To evaluate the effectiveness of the countermeasure, we construct a simplified model of the moving receiver. We assume one of the receivers is moving towards one random direction, and the attacker uses the receiver's original position $\mathbf{P}_i^S$ to place their transmitters. We analyzed the effect of the location offset d of one receiver for values between 0 and $1700m$, which can be caused by the defense mechanism. Figure 6 shows the position error ϵ of multilateration result $\mathbf{x}$ and target position $\mathbf{z}^g$.

With the increasing of the offset, the multilateration result error is increasing. There is a maximum location offset value required for successful false data injection, which can serve as the reference parameter in countermeasures. However, this method needs some vehicles which increase the cost and the moving pattern may also be observed by the attacker easily.

5.2. Changing Operational Status. We can change the operational status of the receive sites randomly, which makes the attackers have a great possibility of injecting the wrong sites and failed to bypass the monitor. The redundant receivers can be added on the basis of the original receiving station, and the central processing station randomly switches the multilateration receivers used to estimate the location. There need to become more than four receivers to cover a target area. All of them pass ADS-B data to the central process station, but only four of them are used for multilateration. After a certain time period, the operational status is changed again. The time period is related to the attackers power to find out the operational status.

To evaluate this countermeasure, we randomly generate the locations of five receivers. Their locations are in the range of one aircraft's ADS-B communication area. We assume the attacker injects messages to four of them but ignores the existence of the other receiver. However, the ignored receiver could still receive them. We evaluate the position error ϵ under different operational status. As shown in Figure 7, the error is undetectable in Status 1. Attackers may inject successfully when the system is operating under this status. But the error cannot be ignored under other status. If we did not detect the attacker at current period, they could be found after the status changing in the next period.

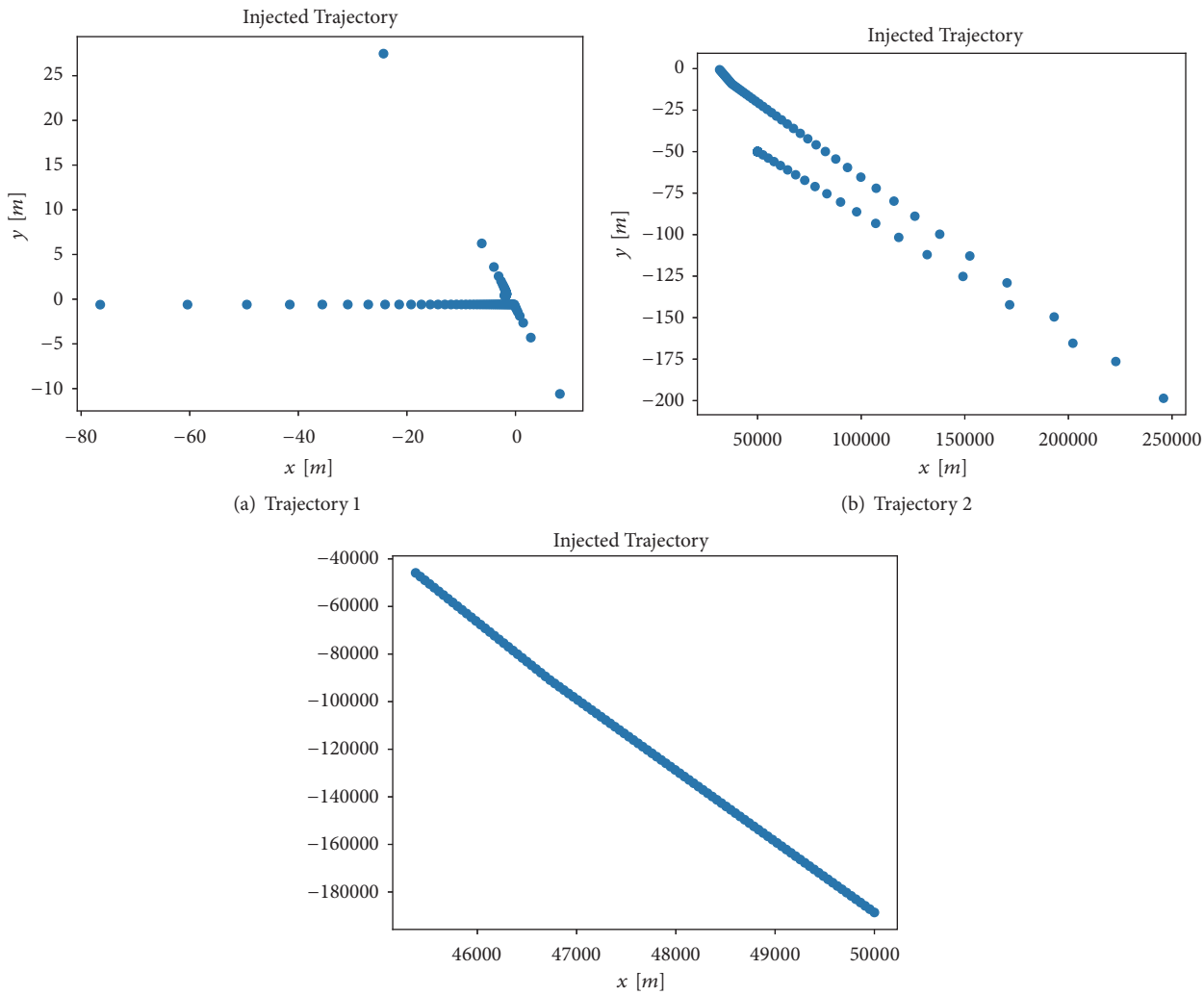
For the defenders, they only need to build 0.25 times more receivers to achieve this. This scheme is much cheaper and stealthier; the attackers have to hack the center process station to get the operational status.

In summary, our countermeasure requires no modifications to the ADS-B signal, the surveillance infrastructure, or the ADS-B receiver; it is resistant against a wide range of attackers, and it can be deployed using multiple standard ADS-B receivers.

6. Conclusions

In this paper, a formal model of the ADS-B multilateration system is established, and a new model of false data injection attack is proposed, which could be used to achieve the optimal attack configuration. This work has shown that a multidevice attacker can inject ghost aircraft with only two transmitters, which makes the cost lower. Finally, we suggest some countermeasures for this attack model. The multidevice false data injection attack model proposed in this paper has a certain generality, which can be applied to other multilateration technology applications. There also exists potential multidevice false data injection threat in the location scenarios based on ZigBee or WiFi technology [32].

In the further study, we will use GNU Radio to perform signal level simulation and try to verify multidevice false data injection attack to bypass the frequency based detection. Second, the above models assume that the location of the attacker's transmitter is fixed. If multiple ghost aircraft tracks are injected, a fixed position attack model may not satisfy the



(a) Trajectory 1

(b) Trajectory 2

(c) Trajectory 3

Figure 5: Spatial analysis of injected trajectories.

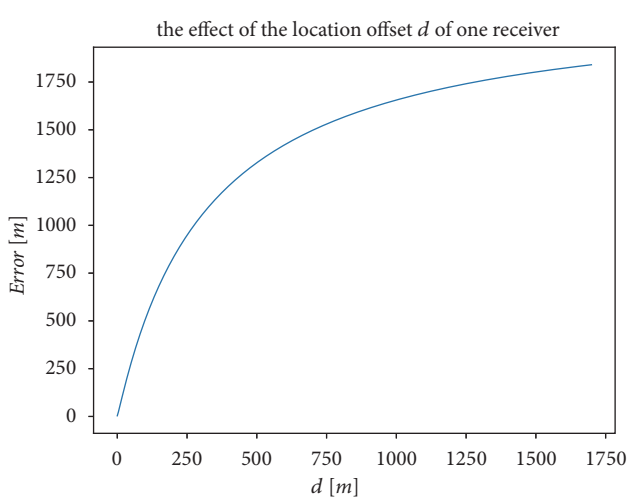


Figure 6: The effect of the location offset d of one receiver.

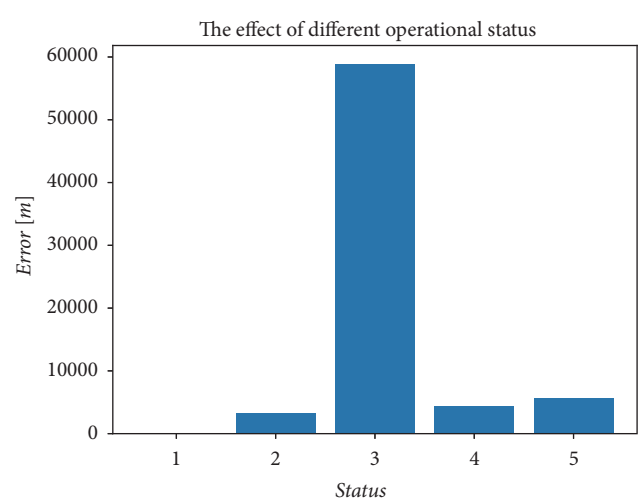


Figure 7: The effect of different operational status.

requirements of false data injection attack, and the attacker's model of moving transmitters is needed to be studied. In addition, we assume the communication graph is complete; i.e., all receivers can receive all transmitted signals. We will study the attack pattern considering the more complicated communication graph. And how to gather information about the ADS-B receivers' locations and operation states to construct the communication graph is also a big question.

Data Availability

The location data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] M. Strohmeier, M. Schäfer, V. Lenders, and I. Martinovic, "Realities and challenges of nextgen air traffic management: the case of ADS-B," *IEEE Communications Magazine*, vol. 52, no. 5, pp. 111–118, 2014.

[2] F. I. Romli, J. D. King, L. Li, and J. P. Clarke, "Impact of automatic dependent surveillance-broadcast (ADS-B) on traffic alert and collision avoidance system (TCAS) performance," in *Proceedings of the AIAA Guidance, Navigation and Control Conference and Exhibit*, p. 6971, USA, August 2008.

[3] D. McCallie, J. Butts, and R. Mills, "Security analysis of the ADS-B implementation in the next generation air transportation system," *International Journal of Critical Infrastructure Protection*, vol. 4, no. 2, pp. 78–87, 2011.

[4] B. S. Ali, "System specifications for developing an automatic dependent surveillance-broadcast (ADS-B) monitoring system," *International Journal of Critical Infrastructure Protection*, vol. 15, pp. 40–46, 2016.

[5] A. Costin and A. Francillon, *Ghost in the Air(Traffic): On Insecurity of ADS-B Protocol and Practical Attacks on ADS-B Devices*, 2012.

[6] Y. T. Chan and K. C. Ho, "A simple and efficient estimator for hyperbolic location," *IEEE Transactions on Signal Processing*, vol. 42, no. 8, pp. 1905–1915, 1994.

[7] R. O. Schmidt, "A new approach to geometry of range difference location," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-8, no. 6, pp. 821–835, 1972.

[8] J. A. Besada, G. De Miguel, A. M. Bernardos, and J. R. Casar, "Automatic-dependent surveillance-broadcast experimental deployment using system wide information management," *International Journal of Microwave and Wireless Technologies*, vol. 4, no. 2, pp. 187–198, 2012.

[9] M. Monteiro, A. Barreto, R. Division et al., "Detecting malicious ADS-B broadcasts using wide area multilateration," in *Proceedings of the 2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, pp. 4A3-1–4A3-12, Prague, Czech Republic, September 2015.

[10] K. Jansen and C. Pöpper, "Advancing attacker models of satellite-based localization systems: the case of multi-device attackers," in *Proceedings of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks - WiSec '17*, pp. 156–159, ACM Press, Boston, Mass, USA, July 2017.

[11] D. Moser, P. Leu, V. Lenders, A. Ranganathan, F. Ricciato, and S. Capkun, "Investigation of multi-device location spoofing attacks on air traffic control and possible countermeasures," in *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking - MobiCom '16*, pp. 375–386, ACM Press, New York, NY, USA, October 2016.

[12] R. SC186, "Minimum operational performance standards for 1090 MHz automatic dependent surveillance–broadcast (ADS-B)," RTCA DO-260, 2000.

[13] R. Do, *Minimum Operational Performance Standards for 1090 MHz Extended Squitter Automatic Dependent Surveillance–Broadcast (Ads-B) and Traffic Information Services–Broadcast (Tis-B)*, vol. 260, Radio Technical Commission for Aeronautics, 2009.

[14] R. DO, *Minimum Operational Performance Standards for 1090Mhz Extended Squitter ADS B and TIS-B*, RTCA, Washington, DC, USA, 2009.

[15] L. Purton, H. Abbass, and S. Alam, "Identification of ADS-B system vulnerabilities and threats," in *Australian Transport Research Forum*, pp. 1–16, Canberra, 2010.

[16] M. Strohmeier, V. Lenders, and I. Martinovic, "On the security of the automatic dependent surveillance-broadcast protocol," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 1066–1087, 2015.

[17] J. Baek, E. Hableel, Y.-J. Byon, D. S. Wong, K. Jang, and H. Yeo, "How to protect ADS-B: confidentiality framework and efficient realization based on staged identity-based encryption," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 3, pp. 690–700, 2017.

[18] M. Strohmeier, V. Lenders, and I. Martinovic, "Lightweight location verification in air traffic surveillance networks," in *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security (CPSS '15)*, pp. 49–60, Singapore, April 2015.

[19] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun, "On the requirements for successful GPS spoofing attacks," in *Proceedings of the 18th ACM conference on Computer and communications security (CCS '11)*, pp. 75–86, ACM Press, Chicago, Ill, USA, October 2011.

[20] D. Sriraman, R. K. S. Kumar, and S. V. Subhashini, "Air traffic controller using a new ads-B framework," 2016.

[21] J. Herrero, J. Portas, F. Rodriguez, and J. Corredera, "ASDE and multilateration mode-S data fusion for location and identification on airport surface," in *Proceedings of the 1999 IEEE Radar Conference. Radar into the Next Millennium*, pp. 315–320, Waltham, Mass, USA, 1999.

[22] J. Johnson, H. Neufeldt, and J. Beyer, "Wide area multilateration and ADS-B proves resilient in Afghanistan," in *Proceedings of the 2012 Integrated Communications, Navigation and Surveillance Conference (ICNS)*, pp. A6-1–A6-8, Herndon, Va, USA, April 2012.

[23] L. Gomez and I. T. Sierra, "Implementation of automatic dependent surveillance (ADS-B) in Colombia," in *Proceedings of the 2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, pp. 2B2-1–2B2-9, Prague, Czech Republic, 2015.

[24] T. Li and B. Wang, "Sequential collaborative detection strategy on ADS-B data attack," *International Journal of Critical Infrastructure Protection*, vol. 24, pp. 78–99, 2019.

[25] M. Schafer, M. Strohmeicr, M. Smith, M. Fuchs, V. Lenders, and I. Martinovic, "OpenSky report 2018: assessing the integrity of crowdsourced mode S and ADS-B data," in *Proceedings of the 2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC)*, pp. 1–9, London, UK, September 2018.

[26] K. Jansen, M. Schafer, D. Moser, V. Lenders, C. Popper, and J. Schmitt, "Crowd-GPS-Sec: Leveraging crowdsourcing to detect and localize GPS spoofing attacks," in *Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP)*, pp. 1018–1031, San Francisco, Calif, USA, May 2018.

[27] M. Strohmeier, V. Lenders, and I. Martinovic, "Intrusion detection for airborne communication using PHY-layer information," in *Detection of Intrusions and Malware, and Vulnerability Assessment*, vol. 9148 of *Lecture Notes in Computer Science*, pp. 67–77, Springer International Publishing, Cham, Germany, 2015.

[28] C. Laurendeau and M. Barbeau, "Probabilistic localization and tracking of malicious insiders using hyperbolic position bounding in vehicular networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2009, Article ID 128679, 13 pages, 2009.

[29] Q. Wang, H. Su, K. Ren, and K. Kim, "Fast and scalable secret key generation exploiting channel phase randomness in wireless networks," in *Proceedings of the 30th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '11)*, pp. 1422–1430, Shanghai, China, April 2011.

[30] D. Steinmetzer, M. Schulz, and M. Hollick, "Lockpicking physical layer key exchange: weak adversary models invite the thief," in *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec '15)*, 11, 1 pages, ACM, New York, NY, USA, 2015.

[31] S. Gao, L. Xie, A. Solar-Lezama, D. Serpanos, and H. Shrobe, "Automated vulnerability analysis of AC state estimation under constrained false data injection in electric power systems," in *Proceedings of the 54th IEEE Conference on Decision and Control, CDC 2015*, pp. 2613–2620, Japan, December 2015.

[32] E. Yaksel, H. R. Nielson, F. Nielson, M. Fruth, and M. Kwiatkowska, "Optimizing zigbee security using stochastic model checking," http://arxiv.org/abs/1205.6675.

Hindawi
Security and Communication Networks
Volume 2019, Article ID 4614721, 35 pages
https://doi.org/10.1155/2019/4614721

Research Article

A Comparative Study of JASO TP15002-Based Security Risk Assessment Methods for Connected Vehicle System Design

Yasuyuki Kawanishi,[1,2] Hideaki Nishihara,[1] Daisuke Souma,[1] Hirotaka Yoshida,[1] and Yoichi Hata[2]

[1] *SEI-AIST Cyber Security Cooperative Research Laboratory, National Institute of Advanced Industrial Science and Technology (AIST), Osaka, Japan*
[2] *Cyber-Security R&D Office, Sumitomo Electric Industries, Ltd., Osaka, Japan*

Correspondence should be addressed to Hirotaka Yoshida; hirotaka.yoshida@aist.go.jp

Received 6 July 2018; Revised 12 October 2018; Accepted 11 December 2018; Published 3 February 2019

Guest Editor: Erwin Schoitsch

Copyright © 2019 Yasuyuki Kawanishi et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In recent years, much attention has been paid to autonomous vehicles and security threats on such vehicles have become an important issue. One of these examples is a command injection issue on a gateway ECU, which was reported in 2016. In order to mitigate these threats, the secure design of connected vehicle systems, which is done at the concept phase during development, has become increasingly important in industry. From this perspective, a security guideline such as JASO TP15002 which specifies two concrete methods, CRSS (CVSS Based Risk Scoring System) and RSMA (Risk Scoring Methodology for Automotive System), was made public in 2015. The latest work on the application of TP15002 to the ITU-T X.1373 standard was published in 2017. However, the risk assessment in this publication seems limited. It is not clear from this publication how systematically the risk assessment task in TP15002 can be performed at the implementation level. Another interesting question is how different methods affect the risk scores of connected vehicle systems. In this paper, we focus on the risk assessment phase in JASO TP15002. For a systematic risk assessment, we introduce an idea of asset container and propose to extend CRSS to a novel RSS (Risk Scoring System), RSS-CVSSv3, by appropriately replacing CVSSv2 vulnerability scoring system on which CRSS is based with CVSSv3. To address the above questions, we perform a comparative study on CRSS, RSMA, and RSS-CVSSv3 for multiple use cases such as a CGW (Central Gateway) and a drone, to examine the efficiency and usefulness of our methods. For this comparative purpose, we devise an interesting approach for the refinement of RSMA to the obstacles in comparing CRSS with RSMA.

1. Introduction

1.1. Positioning of This Paper. This paper is an extended version of the paper that appeared in the proceedings of the DECSoS 2017 Workshop [1]. This extended version includes about 200% new material. The main changes and increments are as follows:

(i) Revision of the abstract

(ii) Revision of the introduction for better description

(iii) Addition of preliminary work regarding general concepts and JASO TP 15002 based

(iv) Addition of new problems of difficulty in comparing CRSS and RSMA

(v) Addition of new proposals of RSS-CVSSv3 and approach for comparing CRSS and RSMA

(vi) Supplementation of all necessary data for case study 1 on connected car system

(vii) Addition of a new result on a case study on a UAV (Unmanned Aerial Vehicle) system

(viii) Addition of discussions and limitations on our proposals

(ix) Supplemental related work.

1.2. The Background and Our Contributions. In recent years, there has been a significant amount of attention devoted to autonomous vehicles. These vehicles have a variety of purposes for which many systems and devices have been developed, ranging from self-driving cars, autonomous robots to aid people in their work in hazardous cities, and UAV which are commonly known as drones.

Recently, it has been known that the security is critical in vehicle systems. Many devices, objects, or entities are highly digitalized and networked in a sense, where vehicle systems can be regarded as Internet-of-Things (IoT) nodes, which allows the users to use various services via Internet. From security point of view, the communication interfaces that allow the vehicles to connect to the Internet can be considered "Entry Points" or "Attack Surfaces" for the attackers to exploit.

Since 2010, researchers have reported attacks on real vehicles, where the attacker can manipulate the control functions in such a way that the car can be controlled in an unintended manner [2, 3]. In 2016, a command injection issue on a gateway ECU, commonly known as the CWE-77 weakness, was reported, for which in CVE 2016-9337 [4] a CVSS Score 6.8, which is computed by means of the formula specified in [5], was assigned.

Regarding the security of the in-vehicle network, in particular, CAN BUS is viewed as a problem [6]. To solve the problem, in [7], the authors studied the new in-vehicle network consisting of several subnetworks, where one subnetwork is connected to another via Central Gateway (CGW).

In addition to the recent hacking of cars, researchers have also found vulnerability in certain types of UAVs (drones), which exploited information security vulnerability in 2015. The attackers took control of driving systems by drones by exploiting sensor systems [8] or interfering with flying functionality by means of attacks via Wi-Fi communications.

For UAVs, although some vulnerabilities were pointed out even before this case [9–11], it is still important to pay attention to these vulnerabilities because UAVs will be incorporated into a larger system, namely, the surrounding environment in the future. For example, the UAV may connect and share its data with other UAVs and the surrounding environmental system [12], or its flight may be controlled in a way that amateurs do not carelessly violate the security and privacy of others [13].

In response to these security concerns, secure design of connected vehicle systems, which is performed at the concept phase during the development, has become increasingly important in industry. From this perspective, a security guideline which incorporates the risk management procedures of information security has been developed [14].

One particularly useful approach is to measure software attack surfaces, which indicate susceptibility to attack. With respect to the security analysis based on attack surface, model-based metrics have recently been developed [15], dealing with the dependability evaluations and the assurance of critical systems.

Regarding IT (Information Technology) security, the concept of risk has been investigated from various perspectives such as risk management [16–18] and risk analysis (cost assessment) [19]. IT security risk assessment is discussed with respect to asset identification, lifecycle clarification, system model definition, and conceptual frameworks for risk assessment [20–23]. There is a document describing the Common Evaluation Methodology (CEM) for Information Technology Security Evaluation, which helps evaluators learn how to apply the CC (Common Criteria) when they perform formal evaluations. ISO/IEC 15408 [24] is based on the security evaluation standard developed by the CC project.

In the automotive industry, several security guidelines and methods have been proposed such as SAE J3061 [25]. JASO TP15002 [26, 27] also deals with high-level methods, as explained in a sequel. In this guideline, unlike in the previous phases, namely, model definition and threat identification, the risk assessment phase has two concrete methods, CRSS (CVSS Based Risk Scoring System) and RSMA (Risk Scoring Methodology for Automotive System). CRSS is a risk evaluation criterion based on CVSS v2 (Common Vulnerability Scoring System), where two metrics Attack Ease and Effect Factor are assigned for each identified threat. RSMA computes the risk score by referencing the Risk Level Decision Table, which consists of Occurrence Possibility and Effect Factor.

To the best of our knowledge, there has been no prior work on these methods in a comparative way, nor on which use case (in one of) these methods work reasonably.

For the last few years, Internet-connected services used in modern vehicles such as cars and drones have received a lot of attention from the public. However, in response to the recent hacking of such services, such as the exploitation of information security vulnerability, the secure design of connected vehicle systems, which is performed at the concept phase during the development, has become increasingly important in industry. From this perspective, a security guideline, JASO TP15002, was made public in 2015. The risk assessment methodology described in TP15002 is applied to the ITU-T X.1373 standard that was published in 2017, dealing with an important security use case, namely, remote software update for vehicles. However, the risk assessment of this work seems limited. The main issues are as follows:

(a) It is not clear how systematically the risk assessment task in TP15002 can be performed at the implementation level.

(b) Another interesting question is how different methods affect risk scores of connected vehicle systems. One of the obstacles is the difficulty in comparing CRSS and RSMA.

In this paper, we focus on the risk assessment phase in JASO TP15002. The importance of JASO TP15002 can be explained by the fact that the significant time and effort of a large automotive committee in Japan have been spent as well as their consensus on what has to be taken into account in the product design phase of the automotive development. Another explanation on its importance can be supported by the fact that the methodology of TP15002 is applied to recently developed ITU-T standard X.1373 [28] that addresses one of the most important use cases, over-the-air software

updates for the vehicle devices. As a result, TP15002 identified 17 classes of threat on the model of the mobile vehicle gateway.

For a systematic risk assessment, we introduce the idea of an asset container and propose to extend CRSS to a novel Risk Scoring System (RSS)s, RSS-CVSSv3, by appropriately replacing the CVSSv2 (Common Vulnerability Scoring System version 2) vulnerability scoring system on which CRSS is based with CVSSv3. To address the above issues in an objective manner, we perform a comparative study on CRSS, RSMA, and RSS-CVSSv3 for multiple use cases such as a CGW and a drone, to examine the efficiency and usefulness of our methods. For this comparison purpose, we devise an interesting approach of refinement of RSMA to the obstacle in comparing CRSS and RSMA.

As a result, we provide some interesting data which could serve as information for better understanding of original and proposed risk assessment methods. We expect that our research will offer an information basis on which future system designers can choose the most suitable security risk assessments for designs of connected vehicles.

The analysis of our method is efficient enough to minimize reworking the guideline. This saves time by using a format which organizes items which allows an engineer to just write a simple description without any ambiguity.

The Need from Industry. This work is motivated by a need from industry with respect to the implementation of secure system design based on a high-level security guideline, such as JASO TP15002. This need is summarized by conducting a secure system design with appropriate efforts. With a naive implementation of JASO TP15002, the following problems require more effort to keep the quality of the result:

(i) There are nondetailed matters in JASO TP 15002. Thus individual designers have to refine or instantiate these matters in their own ways.

(ii) Typically a very large number of threats are identified, which cannot be managed by a designer.

(iii) The quality of output, such as granularity, the amount of description, or validity depends on individual designers.

The Industrial Impact. Our work contributes to the secure development of devices such as a CGW, telematics, or end ECUs that play key roles in Over-The-Air (OTA) applications. This is supported by the evidence that JASO TP15002 is used as a risk assessment methodology in the on-going ITU-T development "secure software update capability for intelligent transportation system communication devices" which was discussed in the UNECE World Forum for Harmonization of Vehicle Regulations (WP.29) forum in 2016 [29]. Consequently, we expect that our work will help in reducing efforts for secure system design, with sufficient quality.

Our Contribution. Our first contribution in this paper is for engineering. For system designers who aim to comply with JASO TP15002, we propose the notion of an *asset container* which leads to a systematic way for secure design, which is consistent over phases 1, 2, and 3, in JASO TP15002. Our approach is to identify gaps between the descriptions in JASO TP15002 and the actual design tasks. Our second contribution in this paper is scientific. We propose a risk assessment method RSS-CVSSv3 based on the well-known vulnerability scoring system CVSS v3 for a reference to future update of JASO TP15002. To pursue our interest in comparing risk assessment methods included in this guideline, we provide an approach to refine RSMA to make the discussion of our risk assessment methods more concrete and more comparative, we focus on particular types of vehicle systems, such as connected cars employing CGW and UAVs.

1.3. Related Work

1.3.1. Vulnerabilities in Cars. Although applying Information Technologies accelerates the evolution of automobiles, it also provides attackers with opportunities to exploit automobiles. As mentioned in [30], implementing intelligent features, especially, like lane keeping assist, may allow critical components, like the steering system, to be controlled from the outside.

In 2010, experimental attacks on actual cars were reported [2, 3]. In those reports, external devices were wired to the in-car network. Attackers could access the network directly; they analyzed traffic and injected fake messages in order to cause irregular behaviors on the car. Furthermore, if the attacker can physically access the car, the attacks are even easier to make. For example, a denial of service attack called the bus-off attack is one of such attacks. Attackers have to use sequences of signals according to the specification, when the car is not physically accessible [31]. Attackers can attack by sequences of signals not according to the specification if they access a car physically [32, 33].

The authors of [2] and [3] discussed remote attacks on actual cars in [34] and [30], respectively. In [30], vulnerabilities in the telematics of a car system and exploits to them were shown in detail. In this case, an attacker could access the telematics easily and send commands to equipment in the car via wireless communication. In 2016, an attack on an actual car was reported by finding a vulnerability assigned CVE 2016-9337 [4] of the Command Injection, for which the CVSS v3.0 Base Score, equal to the RSS-CVSSv3 score, was 6.8. The relevant module was the vehicle's gateway ECU, which our Case Study 1 dealt with in Section 6. Due to this vulnerability, an attacker installed malicious software by sending messages to the vehicle's CAN BUS.

With a fake Wi-Fi access point, attackers accessed the target car system, and then they exploited vulnerabilities in the browser [35]. They were then able to control the car even when it was running. The vulnerabilities allowing a remote attack via a cellular network Telematics Control Unit, when present on actual vehicles produced in a certain period, were reported. For one of these vulnerabilities assigned CVE 2018-9318 [36], the CVSS v3.0 Base Score, equal to RSS-CVSSv3 score, is very high, 9.8.

Attacks on specific functionalities of cars using close-range wireless communication have also been revealed. For instance, a car immobilizer is a system that typically requires the presence of a key fob to allow a car to run. An attack where the attacker was able to track car keys by exploiting

vulnerabilities in a car immobilizer was reported in [37] in 2012. Cracking car immobilizer systems damages the functionalities and a cracked car system may run without the owner's key or stop suddenly on a high way [38]. Security concerns in Tire Pressure Monitoring Systems (TPMSs) were reported in [39]. In this case, when messages are communicated, neither authentication nor input validation can take place.

1.3.2. Vulnerabilities in Drones. For many years it has been clear that drones categorized as Unmanned Aerial Vehicles (UAVs) have been used for military purposes of surveillance, but currently drones have become widely known to the public. Companies have given much easier access to drones than before.

In 2017, researchers pointed out that certain drones sold in the market are vulnerable [8].

Consequently, US-CERT summarized these vulnerabilities and informed the public that certain types of drones allow full file permissions to anonymous users by providing FTP access over its own local access points [40].

The vulnerability note VU#334207[40] and CVE-2017-3209[41] assigned to it can be used to achieve several goals, including that the attackers intentionally cause accidents on a drone. In 2015, in Seattle, an accident of this kind actually occurred [42].

1.3.3. Standardization Activities and Security Guidelines. For the automotive industry, several security guidelines and methods have been proposed. SAE J 3061 [25] is a high-level guideline which includes a definition of the lifecycle process, information for existing tools, and methods.

OCTAVE Allegro [43] is a process-driven threat assessment method. It includes methods for threat identification, risk assessment, and the selection of mitigation approaches. The EVITA method [44] refines abstract attacks into concrete attacks through the construction of Attack Trees and the evaluation of the risks of abstract attacks using those Attack Trees. In the EVITA methodology, risk is evaluated by severity, which includes 4 aspects (safety, finance, privacy, and operation) and attack probability based on the framework of Common Criteria (CC) in which system users can specify their security functional requirements.

The term impact in TP 15002 has the same meaning as in EVITA.

Since 2016, there has been an important standardization joint working project between ISO and SAE, namely, ISO/SAE. This standardization activity on cybersecurity engineering, expected to be published as ISO/SAE 21434, adopts the risk-oriented approach. It attempts to specify Cybersecurity Assurance Level (CAL) [45] which indicates the required level of cybersecurity process rigor, though it is now unclear whether CAL will be normative or informative. In the project, the CAL and the methodology to determine the CAL level will be provided for all the systems and devices throughout the entire automotive supply chain.

1.4. Organization. In Sections 2 and 3, we will introduce preliminary work related to this paper. In Section 4, we will identify the problems in previous work and then we propose our method in Section 5. In Sections 6 and 7, we conduct case studies on the secure designs of publicly available automotive systems. Finally, in Sections 8 and 9, we will discuss our approach and our results and explain the limitations of our work. In Section 10, we present our conclusion.

2. Preliminary 1: Concepts and Framework

2.1. Common Concepts Underlying Safety and Security. We consider that safety is relevant in this paper because the risk score systems we address such as CRSS incorporate criticality of safety as impact factor during their computations. In the automotive industry, it is increasingly important to develop methods for evaluating multiple attributes at the same time and to provide useful information on the entire system. For instance, a methodology for modeling automotive software security, privacy, usability, and reliability has been developed by Ford [46].

The perspectives of CIA (Confidentiality, Integrity, and Availability) of security are increasingly important in safety-critical systems and a security-for-safety paradigm is necessary when hazards originate from threats. A considerable amount of published work exists in this area, for example [47–49].

Automotive systems are typically known as safety-critical systems and the system development process follows the standard ISO 26262 [50], which deals with safety issues. In safety-critical systems, several methodologies of safety analysis and design have been established [51–55], but security was not considered. Various analysis and design methods of security have been proposed in Information Technology [43, 48, 56–62]. Safety-critical systems other than automotive systems have begun to consider the relationship between safety and security. In railway control systems, security affects safety and economy [63]. This relationship between safety and security is called "security informed safety" [64]. Thus, in railway control systems, security is analyzed from two aspects, unsafe train movement and no train movement. In the avionic security standard DO-326A [65], security is analyzed independently. Security-related activities refer to the results of safety related activities but not vice versa. In the FP7 research project SESAMO (SEcurity and SAfety MOdelling), security and safety are analyzed independently [66]. Security and safety requirements derived from analysis converge on a trade-off analysis. Thus, in the SESAMO, security and safety are considered at the same level and affect each other. In industrial control systems, several approaches for combining safety and security including the above concepts are also considered [67, 68].

Referring to well-established work in related disciplines [48], here we give a solid foundation for safety and security engineering by recognizing the similarities and differences in these disciplines. In [48], security and safety are defined as follows:

(i) Safety is the degree to which *accidental* harm is prevented, reduced, and properly reacted to.

(ii) Security is the degree to which *malicious* harm is prevented, reduced, and properly reacted to.

As SAE J3061 showed some overlaps between safety and security exist. But the differences are not entirely clear. For instance, a malicious attack that compromises the integrity of an ECU could eventually lead to accidental harm caused by an operation parameter change in terms of safety. A more generic quality factor that includes these two quality factors is dependability, which is defined as follows:

(i) Dependability [69] is the degree to which various kinds of users can depend on a work product.

Hazard is a situation that increases the likelihood of one or more related accidents [70]. In ISO 26262, a hazard is defined as a potential source of harm caused by the malfunctioning behavior of the item. Note that this definition is restricted to the scope of this standard. A hazard caused by unintended failure and a threat of attacks by attackers can cause harm to assets. More specifically, hazards and threats are defined as follows:

(i) Hazard is the potential source of harm to an asset due to unintended failure of the system. Note that the harm is typically restricted to humans or the environment.

(ii) Threat is the potential source of harm to an asset due to attackers.

The information models of safety and security engineering are significantly similar. For this reason, safety and security requirements can be analyzed regarding a risk-oriented, asset-based approach that considers the accompanied hazards and threats from which these assets have to be protected.

In the above definitions, safety deals with hazards, while security deals with attacks. In order to establish a link between safety and security, the "security-for-safety" paradigm can be introduced in the sense that threats (e.g., intentional hacking attempts) cause hazards that are a potential source of harm.

2.2. The ISO/IEC 15408 Framework in Comparison with ISO 26262. In the automotive industry, there has been an increasing need to develop a framework in which security functions are used in reference to the information system field where ISO/IEC 15408 plays a key role. The purpose of ISO/IEC 15408 is to establish security objectives and evaluation procedure for a Target of Evaluation (TOE), which is a statement that counters identified threats and satisfies assumptions.

In the context of the evaluation of IT products, ISO/IEC 15408 [24] provides a framework and defines security concepts. This standard uses the term TOE where there are modules employing the assets to be protected. In this paper, after defining a TOE, threat identification is conducted and security objectives regarding the major threats are considered.

Referring to [71], we explain the similarities and differences between the ISO/IEC 15408 TOE and an ISO 26262 item definition. Regarding the ISO 26262 item, most required parts of the reference, the overview and description of ISO/IEC 15408 TOE, have been already described. However, ISO/IEC 15408 TOE is extended with an overview of the included security features such as the CIA (Confidentiality, Integrity, and Availability) perspectives and the functionality of the ISO 26262 item. Afterwards, hazard analysis and risk assessment are performed so that potential hazards and their operational situations are identified and an Automotive Safety Integrity Level (ASIL) is determined. Then safety goals are determined for hazardous events (the combination of a hazard and an operational situation [50]).

3. Preliminary 2: Security Evaluation in JASO TP15002

Since 2012, the Society of Automotive Engineers of Japan (JSAE), Inc., has developed a standard procedure for security system design, yielding the guideline JASO TP15002. Its purpose is to describe standard procedures that define the security functions. It considers the prevention of unauthorized operations of the functions that transmit the information to control ECUs (e.g., Body Domain ECUs) from servers and other devices.

In JASO TP15002, the secure system design process consists of five phases: TOE definition, threat analysis, risk assessment, defined security objectives, and security requirement selection. The first three phases are related to security evaluation, while the remaining phrases are about the treatment of major security risks. Hence the tasks and deliverables for the first three phases are reviewed in this subsection.

Another example of security design is that Firesmith [72] defines an asset-based risk-driven procedure for the identification and analysis of security requirements, which consists of 13 general steps.

Also ISO/IEC 27005 [73], based on the international standard ISO 31000 for risk management [74], specifies a security risk management process consisting of 5 iterative activities with 3 activities conducted in activities other than the design phase of the product. Security processes in this literature start with identification of the target or the environment. The processes extract threats and then evaluate and prioritize them. Finally the processes decide treatment of threats as requirements.

3.1. Phase 1: TOE Definition. This phase clarifies the structure of TOE as a model. This model is produced from the assets to be protected and from Data Flow Diagrams (DFD) that specify data flows between the modules in TOE. Thus, the model shows the network structures and entry points in the TOE. Next, properties to protect the CIA perspectives are assigned for each asset in the modules. It is important that the functions of an automotive system operate as correctly as expected, and therefore integrity or availability should be ensured for functions. Similarly, confidentiality or integrity should be ensured for the information exchanged among devices, and for external central servers as well. Information used in the sequel, such as lifecycles and modules of the TOE, is also specified in this phase. Finally, all information related to security evaluation is formalized and shared.

3.2. Phase 2: Threat Analysis. Threats together with the situations they occur in are listed exhaustively in this phase. First, assumptions are stated for feasible analysis. It can be assumed that events or situations that may occur logically but do not actually happen for some reasons are ignored in analysis. Next, adverse actions are investigated exhaustively. The situation of an adverse action is identified with *5W* perspectives ("Who," "When," "Where," "Why," and "What,"). These perspectives take values from information identified in Phase 1 (see Table 1). For example, the factor "When" is valued in the lifecycle phases of the TOE. For each combination of entry points and assets, possible adverse actions should be considered. Moreover, organizational security policy is stated in this phase. Regulations or policies that should be treated with but are not derived from the deliverables of Phase 1 are identified.

3.3. Phase 3: Risk Assessment. This phase estimates the risk levels for all the threats that have been identified in the previous phase. The result tells us about critical threats we have to treat. Two risk evaluation criteria are mentioned in JASO TP15002, namely, CRSS (CVSS v2 [75] Based Risk Scoring System) and RSMA (Risk Scoring Methodology for Automotive Systems). They are reviewed in the following subsections.

In this risk assessment phase, major threats are analyzed more deeply. Using fault trees, a threat is decomposed into combinations of concrete events or situations step by step. Finally basic events, which are the origins of a threat, are identified. We treat threats of TOE by considering the treatment of basic events (conducted in the next phase).

3.3.1. CVSS Based Risk Scoring System. CRSS is a risk evaluation criterion based on CVSS v2 (Common Vulnerability Scoring System version 2). For each threat identified in Phase 2, two metrics, exploitability and impact, are assigned, and the risk value is computed from them. Exploitability depends on how close an attacker must be to the TOE for the attack, how far the entry point is from the asset logically, and how much an attacker must pass authentication to reach the asset. Impact depends on the severity of the threat. Finally, the risk for a threat is categorized into three levels according to the computed risk value: Level III (Critical), Level II (Warning), and Level I (Caution). An example of a result at Phases 2 and 3 in JASO TP15002 is shown in Table 1.

3.3.2. Risk Scoring Methodology for Automotive Systems. In addition to CRSS, JASE also proposed RSMA (Risk Scoring Methodology for Automotive Systems) in JASO TP15002. RSMA is a method adopting the concept defined by ISO/IEC 27005 [73]. It computes the risk score by referencing the Risk Level Decision Table, which consists of Effect Factor and Occurrence Possibility. More specifically, RSMA has the following procedures:

(1) Based on the attack scenario of the threat, decide on the values of the five metrics in the Likelihood list shown in Table 2 (see Section 6.6), sum up their values, and decide on the Occurrence Possibility level with reference to Table 3.

(2) Classify the asset to be attacked into three categories (Damage Category), and decide on the Effect Factor.

(3) Decide on RSMA Risk Level referring to Table 4.

3.4. Inputs and Outputs at Each Phase in JASO-Based Risk Assessment. The outputs and inputs at each phase in JASO TP15002 and the inputs that we propose in our risk assessment proposals and their relations are shown in Table 6, in Table 5, and in Figure 1, respectively.

4. Observations and Problems

4.1. A Concept Model of JASO TP15002. We have reviewed the concepts in JASO TP15002. Figure 2 shows a partial result; concepts in Phases 1, 2, and 3 are identified and related transversely. All concepts are classified into three groups depicted as packages. The first package *system-related information* represents information about the entire vehicle system. Concepts in this package are the inputs to the TP15002 process. *System outline* consists of three concepts: *system configuration*, *system functions*, and *information that the system handles*. The second package *TOE system* represents information actually dealt with in the TP15002 process. Concepts in this package are determined by referring to concepts in *system-related information*. *Component of the TOE system* is a part of *TOE system*, including *entry point* as a subconcept. *Information flow* is a part of *TOE system* as well. *Module function* is related to *asset to be protected* and refers to *component of the TOE system*. The third package *threat information* represents the result of risk assessment in TP15002. *Threat* is determined with the help of concepts in *TOE system*. *Attack Ease (AE)* and *Effect Factor (EF)* are calculated by referring to *threat*, and they determine *risk value* as a result. *Cause* is analyzed for *threat* when required.

Our model makes it clear that there is in-dependency of *lifecycle* from *TOE model*. In Phase 2, the lifecycle contributes to the perspectives of "Who" and "When" and thus can be dealt with independently.

4.2. Observations and Problems with Previous Work. We made the following observations and identified the following problems with each phase.

4.2.1. Observation in JASO TP15002 Phase 1. Vehicle systems are typically complex. For instance, a whole car employs up to 70 ECUs, which are connected via various protocols such as CAN. It is possible that the in-vehicle devices are developed by different suppliers and vendors. Therefore, the way to form a TOE model depends on them. This means that the TOE could be very different and the model may not be elaborated in a suitable manner. Since a TOE model consists of just functions (and assets in them) and information flows, systems dealing with them in vehicles can be TOEs in general. For instance, it is reasonable that car manufactures consider the whole car as the TOE. However, some suppliers consider only a group of in-vehicle devices responsible for infotainment as the TOE. We observe that, even when there are two TOEs (the whole vehicle and some parts of it) for the same vehicle, it is

Table 1: An Example of a Result at Phases 2 and 3 in JASO TP15002.

#	Where	Who	When	Why	What	AE	EF	Risk Value
1	OBD-II	Outsider	In regular use	Maliciously	cause malfunction	3.9	9.2	6.6

Table 2: IN_{3-5}: Occurrence Possibility for RSMA.

Metric	Rank	Value	Definition
Time required	Practical(P)	0	The time required for the attacker to identify and exploit a vulnerability is practical.
(T)	Unrealistic(U)	19	The time required for the attacker to identify and exploit a vulnerability is unrealistic.
Expert knowledge	Nonprofessional(N)	0	Technical expertise is not needed.
(E)	Professional(P)	3	Technical expertise is needed.
TOE knowledge	Open information(O)	0	TOE knowledge is open information.
(TOE)	Limited(L)	3	TOE knowledge is the information that dealers, developers, and manufacturers can obtain.
	Closed(C)	7	TOE knowledge is the information that only a limited number of persons can obtain.
Opportunity	Always(A)	0	The attacker can access TOE unrestricted or the access is not needed.
(O)	Limited(L)	4	The attacker can access TOE but the frequency is limited.
	Impossible(I)	19	The attacker cannot access TOE.
Device	Off-the-shelf(OS)	0	Hardware and software used for attacks are the off-the-shelf products.
(D)	Special(S)	4	Hardware and software used for attacks are the special products.
	Ordered(O)	8	Hardware and software used for attacks are the specially ordered products.

Table 3: Occurrence Possibility Level Assessment Table for RSMA.

Occurrence Possibility Level	Criteria
Large (L)	0 to 14
Medium (M)	15 to 24
Small (S)	25 or higher

important to the outcomes of the security design that these two TOEs are consistent with each other and any interactions have to be clear.

4.2.2. Problems in JASO TP15002 Phase 2. Phase 2 of JASO TP15002 suggests that the 5W ("Who," "When," "Where," "Why," and "What") perspectives should be considered at the same time for each threat. However, we encounter a problem, namely, the difficulty in exhaustively identifying all the possible threat descriptions from the perspectives of "Where" and "What."

We point out that this difficulty originates from ambiguity in the definitions of the 5W characteristics. More specifically, there is a problem with the definition of "What," specifically, what the attacker does. The description for the perspective of "What" could be ambiguous or abstract such as 'breaking the central gateway;' hence, the threat identification (Phase 2) might be performed in an ad hoc manner. This could result in producing useless and unnecessary descriptions (data) of threats for Phase 3. On the other hand, important threats might not be identified as a consequence of this ad hoc search and the entire process might end up with a reworking of Phase 2. The 5W method can identify few hundreds or even thousands of threats. Phase 2 might not be performed in a systematic manner; therefore, an exact description of "What" is unclear. Another problem is that it is unclear how logically and closely this phase task is related to tasks in Phase 1 and Phase 3. More specifically, a question arises as to which metric for risk assessment is related to "Where," "When,"..., or, "What."

4.2.3. Problems in JASO TP15002 Phase 3. CRSS specified in JASO TP15002 is based on the CVSS v2 vulnerability scoring system that was published in 2007. However, CVSS v3 has also been published. Therefore it is necessary to consider whether CRSS is suitable for current or future vehicle systems. RSMA, another method, may be more suitable than CRSS. One of the obstacles in determining a suitable method is the difficulty in comparing CRSS and RSMA.

4.2.4. Problem in Application of JASO TP15002 to ITU-T X.1373. The latest work on JASO TP15002 can be referred to in the ITU-T X.1373 [28] standard that was published in 2017, dealing with an important security use case, namely, remote software updates for vehicles. The standard defines TOE for VMG (Vehicle Mobile Gateway) which is recognized as a key component of a secure software update.

The problem we identified with this work is that the investigation of the risk assessment seems limited. The TOE contains only one module, VMG, which supports nine functions that include mobile communication function, software get function, remote software update function, GPS reception function, Wi-Fi connection function, USB connection function, CAN communication function, CAN gateway function, and OBD connection function, each of which has one asset or a few assets. The threat analysis result shows the relatively small number of 19 threats. In addition, the risk scores are computed for only 3 threats.

This work is reasonable for the purpose of grasping the idea of how the JASO-defined 3-phase risk methodology works. However, in practice, the system is likely to be more

Table 4: Risk Level Assessment Table for RSMA.

Damage category	Effect Factor (EF)	Definition	Occurrence Possibility		
			Small	Medium	Large
Safety	None	No effect on humans	0	0	0
	Small	Mild	L	L	M
	Medium	Severe	L	M	H
	Large	Life-threatening	M	H	H
Personal information/privacy	None	Not personal information/privacy	0	0	0
	Small	A single piece of information cannot identify the individual	L	L	M
	Large	Information that can identify the individual	M	M	H
Property/corporate value	None	No impact on property/corporate value	0	0	0
	Small	Impact only internally (impact on business: small)	L	L	M
	Medium	Impact on customers (impact on business: medium)	L	M	H
	Large	Impact on customers and business (impact on business: large)	M	H	H

Table 5: The Inputs at Each Phase Defined in JASO TP15002 and Our Proposed Inputs(IN_{3-7},..., IN_{3-10}).

Phase #	Label	Input in the corresponding phase
1	IN_{1-1}	System configuration
1	IN_{1-2}	System function
1	IN_{1-3}	Information/data that the system handles
2	IN_{2-1}	System Preliminary condition
2	IN_{2-2}	System outline
2	IN_{2-3}	TOE Model
2	IN_{2-4}	Module Function List
2	IN_{2-5}	Lifecycle List
3	IN_{3-1}	TOE Model
3	IN_{3-2}	Module Function List
3	IN_{3-3}	Attack Ease for CRSS
3	IN_{3-4}	Effect Factor for CRSS
3	IN_{3-5}	Occurrence Possibility for RSMA
3	IN_{3-6}	Effect Factor for RSMA
3	IN_{3-7}	Exploitability for RSS-CVSSv3
3	IN_{3-8}	Impact for RSS-CVSSv3
3	IN_{3-9}	Occurrence Possibility for Q-RSMA
3	IN_{3-10}	Effect Factor for Q-RSMA

Table 6: Output at Each Phase in JASO TP15002.

Phase #	Label	Output in the corresponding phase
1	OUT_{1-1}	TOE Model
1	OUT_{1-2}	Module Function List
1	OUT_{1-3}	Life-cycle List
2	OUT_{2-1}	Assumptions List
2	OUT_{2-2}	Threat List
2	OUT_{2-3}	Organizational security policies List
3	OUT_{3-1}	(Prioritized) Threats List
3	OUT_{3-2}	FT

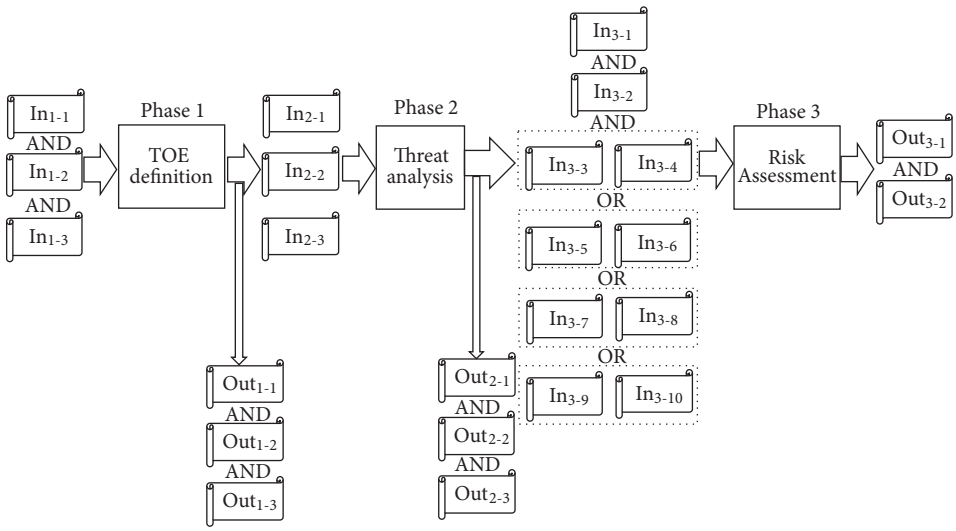


Figure 1: Relation of inputs and outputs including both defined by JASO and the ones that we propose.

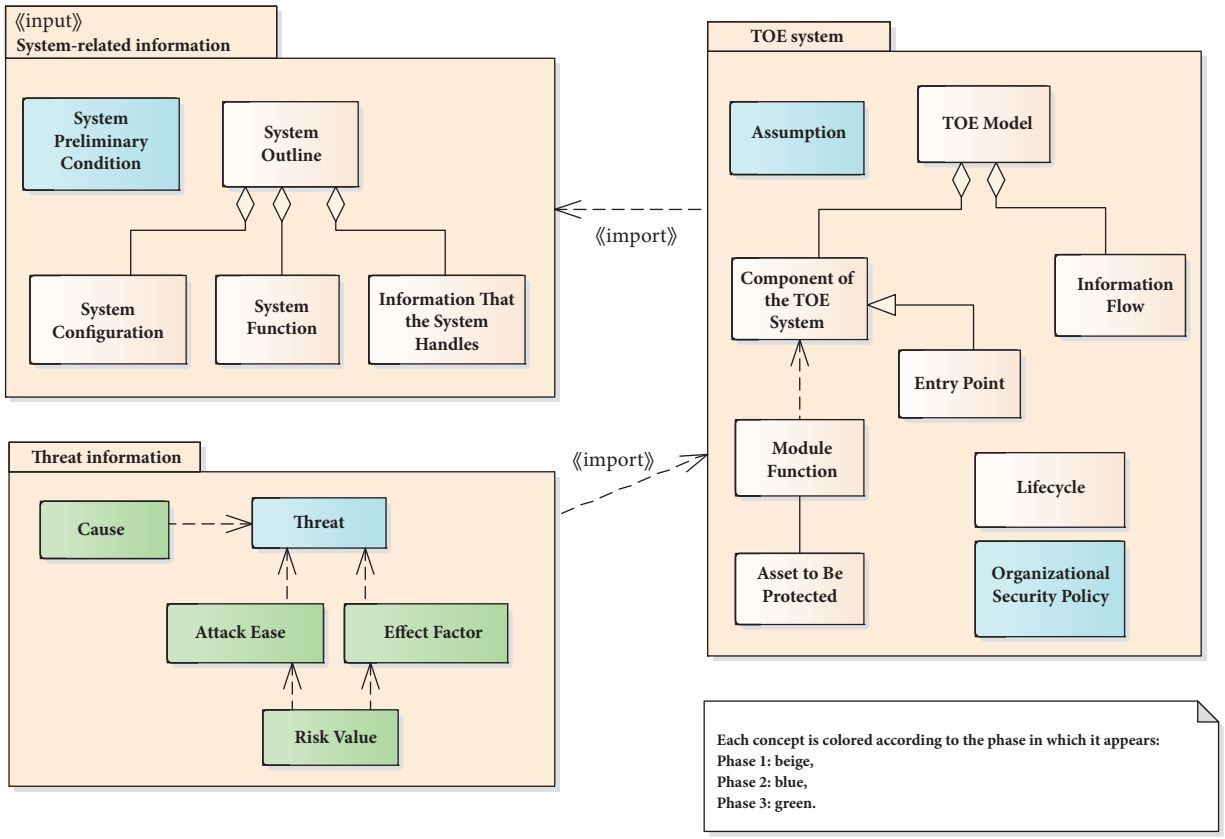


Figure 2: Logical relations in JASO design process (Phase 1 to Phase 3).

complex. There is an important gap between this work and a thorough study. It is not clear how systematically the risk assessment task in TP15002 can be performed at the implementation level. Another interesting question is how different methods affect the distribution of risk scores for the same connected vehicle system.

4.3. A Taxonomy Observation of the JASO Design Process. Our analysis identified a problem with the JASO design process by investigating the well-known dependability taxonomy dealing with safety and security described in Section 2 based on [48]. As mentioned there, some security threats cause failure in the target system, which is a concern in

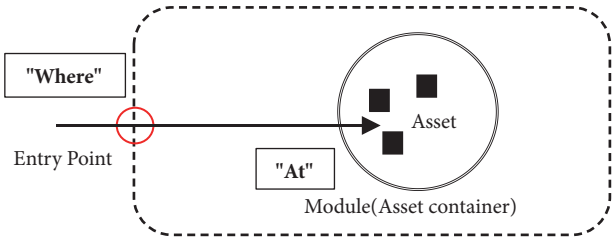


Figure 3: Our idea of an asset container.

safety. Indeed, in our case study in Section 6, the threat that control function in Chassis module suffers from a malware is identified. This threat can easily cause an accident in a vehicle.

JASO TP15002 describes the CIA perspectives of security but it does not show a link to safety. In addition, although JASO TP15002 focuses on security, its method allows us to pick up safety-related threats. Hence it is worth distinguishing safety-related threats in threat analysis, and letting safety evaluation refer to the result.

5. Our Proposed Methods

In this section, we propose our methods for solving the problems stated in the previous section. Our methods include an efficient and systematic way of implementing the 3-phase risk assessment methodology specified in the JASO TP15002 guideline.

5.1. Proposed Notion: An Asset Container for a Systematic Approach in Phases 2 and 3. To solve the problem stated in the previous section, we propose a new description for "What" using "At" (attack destination) and "Asset" (the targeted asset) at the threat analysis phase (Phase 2). Our objective is to improve the efficiency of the *total* work of Phase 2 (threat identification) and Phase 3 (risk evaluation).

Our idea behind this approach is that the Phase-2 work identifies threats in a systematic manner and this work only outputs a list of threats whose description contains information necessary and useful for the risk-score computation in Phase 3.

The perspective of "Where" represents one of the entry points in JASO TP15002, but, in our approach, the perspective of "What" describes what asset of what asset container is attacked. Next, we define the asset container as a module that contains the assets to be targeted as the "At" perspective (see Figure 3).

Therefore, it is important to distinguish these two perspectives strictly from the beginning of Phase 2. On the other hand, only two perspectives, "Where" and "What"(= "At" + "Asset") are necessary to cover all threats, and the remaining three are not necessary up to Phase 3. Though a division of "What" into "At" and "Asset" is added, the number of perspectives for risk analysis drops from five to three, so the workability will be increased.

In our paper, by considering the interdependency we described in the previous section, we propose a 2-step threat identification as follows.

Step 1. Exhaustively identify threats, each of which describes the following three perspectives: "Where," "At," and "Asset."

Step 2. For the identified threats, identify associated threats each of which adds descriptions of the following three perspectives: "Who," "When," and "Why."

Step 1 refers to the TOE model and the module function list, both of which have been produced at Phase 1 in the JASO design process. Step 2 refers to the lifecycle list, which has been produced at Phase 1 in the JASO design process.

After applying the above two steps, our method outputs information shown in Table 7 which includes the data produced by Phases 2 and 3 in JASO TP 15002.

Advantage of Our Method. Our method has three advantages over a naive implementation of JASO:

(i) **From a working procedural perspective**, we make progress in formalizing a security evaluation job. Our method can, especially, separate this job into two tasks, thus enabling an optimization of this job in the sense that our method can refactor in factors necessary for risk assessment.

By introducing an asset container which covers "At" and "Asset" in addition to the"What" perspective, all intrusion routes are *clearly and systematically* identified and all attack targets can be grasped without exception.

(ii) **From a delivery perspective**, our method can improve the quality of the delivery (output) of security because we can remove ambiguity from the threat descriptions.

An advantage may exist that encompasses all threats and an accurate computation of risk scores for each can be realized.

(iii) **From an operational perspective,** we propose our method as a practical technique that makes clear what the developer and the factory operator who are not necessarily security specialists should keep in mind.

5.2. Proposed Method: RSS-CVSSv3. We propose a novel risk assessment method, RSS-CVSSv3: The risk scoring system used CVSS version 3 in place of CVSS version 2 of CRSS.

5.2.1. The Idea Inspired by the CRSS Construction. We take a close look at CRSS based on CVSS metrics. We notice that this method makes a clear distinction between what the system designers can take into account during the concept phase and what has to be taken into account after the release of the product to the market. Our view is that this is why CRSS metrics only include six CVSS v2 base metrics (AV, AC, Au, C, I, and A) that are reasonably considered during the concept phase while these metrics exclude CVSS v2 metrics categorized as three temporal or five environmental metrics. We observe that this distinction comes from the important difference between what risk assessment is in the context of security design and what vulnerability analysis is. We note

Table 7: An Example of Data Produced by Our Method.

#	Where	Who	When	Why	What (At, Asset)	AE	EF	Risk Value
1	OBD-II	Outsider	In regular use	Maliciously	cause malfunction (CGW, Control function)	3.9	9.2	6.6

Table 8: CVSSv3 Basic Metrics [5] and Changes from CVSSv2.

Metric	Name	Short Description	Changes from CVSSv2
AV	Attack vector	This metric reflects the context by which vulnerability exploitation is possible.	Same metric
AC	Attack Complexity	This metric describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability	Same metric, but rank reduced (3 ⟶ 2).
PR	Privileges Required	This metric describes the level of privileges an attacker must possess before successfully exploiting the vulnerability.	Expanded from Au (Authentication), and rank reduced (3 ⟶ 2).
UI	User Interaction	This metric captures the requirements for a user, other than the attacker, to participate in a successful compromise of the vulnerable component.	New metric. Whether a user (other than attacker)'s interaction is necessary for successful exploitation.
S	Scope	refers to the collection of privileges defined by a computing authority (e.g., an application, an operating system, or a sandbox environment) when granting access to computing resources (e.g., files, central processing unit (CPU), memory, etc.).	New metric. Whether the vulnerability spreads to other resources beyond the exploited component.
Im-C	Confidentiality Impact	This metric measures the impact to the confidentiality of the information resources managed by a software component due to a successfully exploited vulnerability.	Same metric, only renamed from EF-C.
Im-I	Integrity Impact	This metric measures the impact to integrity of a successfully exploited vulnerability	Same metric, only renamed from EF-I.
Im-A	Availability Impact	This metric measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability	Same metric, only renamed from EF-A.

that the former is within the scope of CRSS and the latter is within the scope of CVSS.

As mentioned above, JASO TP15002 adopts CRSS as the quantification method for risk assessment at Phase 3. However, CVSS used for CRSS (version 2)[76] has a new version (version 3)[5]. Many security standards or guidelines have been updated to mitigate security attacks. These standards have only got better and new vulnerabilities are reported on a daily basis. It is natural for us to assume that TP 15002 could be updated to address the latest trends of vulnerability in the future.

5.2.2. RSS-CVSSv3 Score Computation. Against this background we propose a novel risk assessment method, namely, RSS-CVSSv3, based on applying the scoring systems, CVSSv3 (Common Vulnerability Scoring System version 3), which is a new system that scores vulnerability in embedded systems and standardized in ITU-T X.1521(03/2016)[5].

The idea is to include only the basic metrics of CVSSv3 shown in Table 8 because our focus is on security design rather than vulnerability analysis. Compared with CVSSv2, there are some changes of metrics in CVSSv3. To illustrate this, Table 8 also shows some differences of the base metrics between the two standards.

"S"(Scope) is a new metric of CVSSv3, which indicates whether the vulnerability spreads to other resources beyond the exploited component, and from which the formula changes a little. As Scope is unchanged normally, the formula of RSS-CVSSv3 is similar to that of CRSS.

The Formula for RSS-CVSSv3: Based on [5], the score *RSS-CVSSv3_Score* is calculated as follows, depending on parameter *S*.

If the Scope is Unchanged(S=U):

$$RSS\text{-}CVSSv3\_Score = \min\left[\left(Impact + Exploitability\right), 10.0\right]$$

$$Exploitability = 8.22 \times AV \times AC \times PR \times UI$$

$$Impact = 6.42 \times [1 - \{(1 - Im\text{-}C) \times (1 - Im\text{-}I) \times (1 - Im\text{-}A)\}] \quad (1)$$

Else if the Scope is Changed(S=C):

$$RSS\text{-}CVSSv3\_Score = \min\left[1.08 \times \left(Impact + Exploitability\right), 10.0\right]$$

$$Exploitability = 8.22 \times AV \times AC \times PR \times UI$$

$$Impact = 7.52 \times \left[ISC_{Base} - 0.029\right] - 3.25 \times \left[ISC_{Base} - 0.02\right]^{15} \quad (2)$$

$$ISC_{Base} = 1 - [(1 - Im\text{-}C) \times (1 - Im\text{-}I) \times (1 - Im\text{-}A)]$$

5.3. Proposed Approach for Comparing CRSS with RSMA. RSMA is a system that evaluates risk based on an attack scenario and the environment of the attacker, whereas CRSS is a system that evaluates risk based on the asset to be attacked. Therefore, in this paper we attempt to compare their results and to analyze the causes of the difference.

As mentioned previously, JASO TP15002 adopts CRSS as the quantitative risk assessment method at Phase 3, while RSMA is not quantitative. Therefore, we need to customize RSMA in a way that we can calculate figures as the assessed value, compare their results, and analyze the cause of the differences.

In the following formula we adopt RSMA methodology to our risk scoring system. Hereinafter we refer to this method as Q-RSMA (Quantifiable-RSMA). In the original RSMA, the Occurrence Possibility for a threat is decided by evaluating the sum of metrics at the three levels. Here we apply the values of metrics directly. Four levels for Effect Factor (None, Small, Medium, and Large) are converted in numerical forms. In the formula, the Effect Factor values (0.0, 10.0, 20.0, and 30.0) and the constant (34.0, and 100.0) are decided so that the value takes between 0 and 10 and their histogram has one peak.

This method uses the multiplication of Occurrence Possibility and Effect Factor, whereas CRSS uses the addition of Attack Ease and Effect Factor. RSS-CVSSv3 also uses the addition of Exploitability and Impact. Q-RSMA might have different values and prioritized threat lists from CRSS and RSS-CVSSv3. There is a difference in multiplication and addition. The reason why we choose different approach is that we are interested in how different the results of different risk scoring systems are for the same TOE. We expect that our work will be a good reference for the designers to choose more appropriate risk scoring systems for their TOE.

The Formula for Q-RSMA:

$$Q\text{-}RSMA\_Score = Occurrence\_Possibility \times \frac{Effect\_Factor}{100.0}$$

$$Occurrence\_Possibility = \max\left[\left(34.0 - \left(Time\_required + Expert\_knowledge + TOE\_knowledge + Opportunity + Device\right)\right), 0.0\right]$$

$$Effect\_Factor = \overline{v}$$

$$\text{where } \overline{None} = 0.0, \quad \overline{Small} = 10.0, \quad \overline{Medium} = 20.0, \quad \overline{Large} = 30.0, \quad (3)$$

$$\text{and where } v \in \begin{cases} \{None, Small, Medium, Large\} & \text{(Damage is Safety)} \\ \{None, Small, Large\} & \text{(Damage is Personal info./privacy)} \\ \{None, Small, Medium, Large\} & \text{(Damage is Prop./corp.value)} \end{cases}$$

Table 9: IN_{3-10}: Effect Factor for Q-RSMA.

Asset Type	Rank	Value	Description
Safety,	Large	30	The assets related to safety or
Property/	Medium	20	the assets related to the added value of the product
corporate value	Small	10	
	None	0	
Personal information/	Large	30	The assets related to the customer's private information.
Privacy	Small	10	
	None	0	

In Table 9 there is an Effect Factor list used for Q-RSMA. The RSMA Effect Factor is numerized in Table 9. This table is added for the quantification of the RSMA result.

6. Case Study 1: A CGW-Employed Connected-Car System

Here we present our results on a case study where the target system is a connected-car system employing CGW as a key component. We apply our method to this case study and then we show the prioritized threats. First, we define the following system specifications of our connected car system:

(i) Functional safety of each ECU is maintained by itself.

(ii) Safety is maintained, even if any of the communications between the ECUs is made impossible.

(iii) The connected-car system is not automated.

6.1. Results of Phase 1: TOE Definition. As an interface to the outside, there is an on-board diagnostics (OBD) connector, a mobile communication module, a global positioning system/global navigation satellite system (GPS) signal reception device, a wireless fidelity (Wi-Fi), a Bluetooth connection, a CAN connection, a universal serial bus (USB) connector, and a secure digital (SD) connector. Based on the vehicle architecture shown in [77], in Figure 4, TOE is defined as the area surrounded by the dotted line and it communicates with the outside of the vehicle via a connection interface.

6.1.1. OUT_{1-1}: TOE Model. (See Figure 4).

6.1.2. OUT_{1-2}: Module Function List. Based on the TOE model, its submodules and assets to be protected are shown in Table 10. As for assets in the CRSS method, their importance is determined from the viewpoints of CIA for security, Confidentiality, Integrity, and Availability.

6.1.3. OUT_{1-3}: Lifecycle List. After writing the details of the TOE model, we write the lifecycle of TOE, and the people involved with this lifecycle. Table 11 shows the lifecycle list.

6.2. Results of Phase 2: Threat Analysis. We present the results on out threat extraction. In this phase, we confirmed the assumptions and conditions of the TOE modeled at Phase 1 and extracted the threats.

6.2.1. OUT_{2-1}: Assumption List. Before performing this evaluation, we assume that no *security functions* are employed in CGW because our position is that the security functions to be employed will be made clear after the implementation of the final phase of JASO TP15002. Our study makes clear how we can prioritize the CGW-related threats, by considering all of them related to the entire vehicle structure, which we take as the TOE model.

Table 12 shows the preliminary assumptions in advancing risk assessment, which also include assumptions other than CGW.

6.2.2. OUT_{2-2}: Threat List. Table 13 is a part of the threat list made by applying the asset container perspectives. We carefully cover all the combinations of the perspectives of "Where," "At," and "Asset," and 'add' or 'remove' threats according to other perspectives (e.g., "Who" or "What") and the assumption list.

In this case study, 145 threats in total have been extracted, including 15 threats related to CGW.

6.3. Results of Phase 3: Risk Assessment in the CRSS Case. In Phase 3, we prioritize these threats by their risk value, which is quantified by the CRSS method.

In advance, we need to set the criteria for each metric used for CRSS, by which each condition gets a numeric value.

6.3.1. IN_{3-3} and IN_{3-4}: Attack Ease and Effect Factor for CRSS. Table 14 shows the metrics and the criteria related to Attack Ease (AE). These AE metrics indicate ease of attack from the viewpoint of the attack route. We allocate a combination of the entry point and the module to the rank of a metric.

Table 15 shows the metrics and the criteria related to Effect Factor (EF). These EF metrics indicate the importance of the asset from the viewpoint of the security CIA and the seriousness of the attack when it is successful.

6.3.2. OUT_{3-1}: Prioritized Threat List. Our study identified 145 threats in total, with the top 16 shown in Table 17. 9 threats are classified as the 'Critical' risk (Level III). 93 threats are classified as the 'Warning' risk (Level II). 43 threats are classified as the 'Caution' risk (Level I). Table 16 shows the definition of each risk level.

This result shows that the more highly evaluated significant asset in the vicinity of the entry point which can be accessed from a remote place is more likely to be attacked.

Table 10: OUT_{1-2}: Module Function List.

#	Module name	Function	Assets to be protected	Confidentiality	Integrity	Availability
1	Power-train	Control functions for driving vehicle related to the Engine, Motor, Fuel,	Control function		√	√
		Battery, Transmission, etc.	Authentication function		√	√
			Authentication information	√	√	
			Sensor information		√	
2	Chassis	Control functions for operating vehicle related to Brake and Steering.	Control function		√	√
3	Body	Control functions for operating vehicle body equipment related to	Control function		√	√
		Door locks, Air conditioners, Lights, and Blinkers.	Control input data		√	
4	ADAS	Automatic brake, Lane-keeping control, Intervehicle distance	Control function		√	√
		control, etc. Functions which bring safety and comfort working	Location information		√	
		together with other vehicle control functions.				
5	CGW	Functions for integration and transformation of the Ethernet and CAN	Authentication function		√	√
	(Central Gateway)	communication. At the same time, this module acts as fault diagnosis	Authentication information	√	√	
		port, OBD (On-Board Diagnostics)-II.	Vehicle status information		√	
6	ITS	Functions via roadside-to-vehicle or vehicle-to-vehicle communication,	Authentication function		√	√
		ETC, ITS (Intelligent Transport System), etc.	Authentication information	√	√	
			Control information		√	√
7	Telematics	Functions for remote control services. For example, collection service	Authentication function		√	√
		for location information, remote door-lock service, remote lighting-on	Authentication information	√	√	
		service, etc.	Personal information	√	√	
			Request information to server		√	√
			Vehicle status information	√	√	
8	Infotainment	Functions for information and entertainment. For example, car	Authentication function		√	√
		navigation systems, audio equipment, key-less entry systems, etc.	Authentication information	√	√	

Table 11: OUT_{1-3}: Lifecycle List.

Phase	Subphase	Overview	Participants
Plan	-	This is the phase in which the planner and the developers of the car's manufacture plan their product. They formulate their car's concept, for example, the scope of user, how to use the services of their car, etc. In addition, they summarize and determine the required specifications of the car which include functional and nonfunctional items. As they evaluate the entire costs of the car system throughout its entire lifecycle at this phase, "the security level" of the system will be decided, as well as how much they can assure the security of their product. It is important for the requirement definitions to include the security requirements and not to include the vulnerabilities.	OEM staff
Development	Product design	This is the phase in which the developers of the car's manufacture and the parts maker design the hardware and the software based on the requirement definitions of the plan phase, in which they plan the implementation of the car.	Supplier staff
	Manufacturing	It is necessary at this phase that "the requirement definitions are correctly implemented in the product," "the vulnerabilities shall not be included in the product," and "the vulnerabilities, if included must be detected before the shipment of the product".	OEM staff Supplier staff
Operation	Shipping	This is the phase in which the user buys the car through the dealer and drives it. For example, in this phase, the status of the car, such as its location, the downloaded software, the operation history of the user, and the driving history of the car, is collected. Also, when the user buys a second-hand car through the second-hand broker, it is necessary for the broker to consider both what should be kept ongoing (e.g., the latest security patch of the firmware) and what must be erased in advance (e.g., the personal information of the previous owner) of the accumulated information.	Carrier staff
	Registration		Dealer staff Second-hand broker
	Regular use, operation	There are many cases in which multiple users who are not the owner of the car use the car in a short term, for example, for car-sharing, car rental, or as a company car. In these cases, it is also necessary to consider the protection of privacy regarding the previous information about the car, its drivers, and types of contract, etc.	Owner/User Outsider
	Maintenance	Furthermore, it is necessary to build a system which allows the users and owners of the car to know its vulnerabilities if some are found after the shipment. This phase also deals with cooperation between the car dealer, the broker, and the maintenance factory.	Administrator of Server Maintenance staff
Disposal	-	This is the phase in which the owner sells or discards the car to replace it or it breaks down. Two cases exist in which the owner will let go of the car: (1) the owner sells the car to another person through a second-hand broker, or (2) the owner deletes the car's registration and throws it away. In this phase certain procedures and confidential data (such as in-person inspection, issuing evidence, etc.) must be confirmed by the owner.	Dismantler Second-hand broker

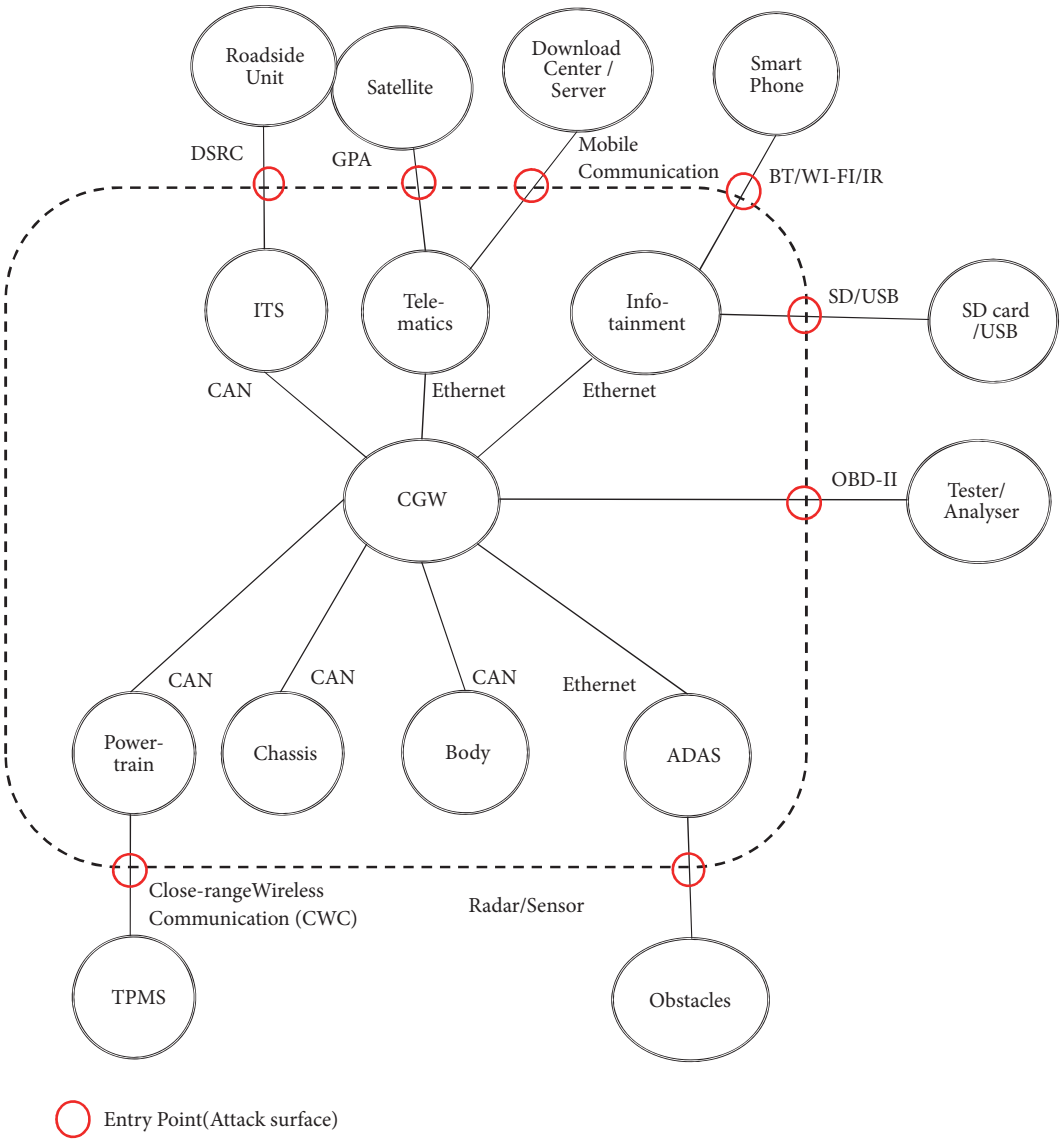


Figure 4: OUT_{1-1}: TOE Model.

Table 12: OUT_{2-1}: Assumption List.

Type	Assumption	Overview
Objective	A.No_Objective_to_CGW	It is assumed that there is no advance security objective for CGW, in order to identify all key threats at the initial stage of security design.
Physical	A.Physical_access_by_Outsider	The "Chassis" module, the "Body" module, and the "ADAS" module are connected to other modules only via an in-car bus, so they are not attacked directly through any physical interfaces.
	A.Attack_to_Sensor_Receptor	The "ADAS" module also has radar and sensors as its interface. These receptors from which this module gets the location information are treated as physical interfaces without transmission function.
Human Factor	A.Automatic_drive	The transfer of the driver's authority is not assumed regarding the use of the functions for automatic drive.
Communication	A.GPA	The communication to Satellite is a one-way (Satellite -> Vehicle), the same as the Sensor.

Table 13: OUT_{2-2}: Threat List.

#	Where	At	Asset	Who	When	Why	What
1	OBD-II	Power-train	Control function	Maintenance staff	In maintenance	Accidentally	cause malfunction
6	CWC (TPMS)	Power-train	Control function	Outsider	In regular use	Maliciously	cause malfunction
10	DSRC (ITS)	Power-train	Control function	Outsider	In regular use	Maliciously	cause malfunction
31	Mobile (Telematics)	Chassis	Control function	Outsider	In regular use	Maliciously	cause malfunction
32	BT/Wi-Fi/IR (Infotainment)	Chassis	Control function	Outsider	In regular use	Maliciously	cause malfunction
34	SD/USB (Infotainment)	Chassis	Control function	Owner/User	In regular use	Accidentally	infect with malware
52	CWC (TPMS)	ADAS	Control function	Outsider	In regular use	Maliciously	cause malfunction
55	DSRC (ITS)	ADAS	Location information	Outsider	In regular use	Maliciously	write wrong data
56	Mobile (Telematics)	ADAS	Control function	Outsider	In regular use	Maliciously	cause malfunction
64	OBD-II	CGW	Control function	Outsider	In regular use	Maliciously	cause malfunction
69	CWC (TPMS)	CGW	Flow/Storage data	Outsider	In regular use	Maliciously	write wrong data
70	DSRC (ITS)	CGW	Control function	Outsider	In regular use	Maliciously	cause malfunction
72	Mobile (Telematics)	CGW	Control function	Outsider	In regular use	Maliciously	cause malfunction
75	BT/Wi-Fi/IR (Infotainment)	CGW	Flow/Storage data	Outsider	In regular use	Maliciously	write wrong data
77	SD/USB (Infotainment)	CGW	Control function	Owner/User	In regular use	Accidentally	infect with malware

Table 14: IN_{3-3}: Attack Ease for CRSS: Value for Rank of Each Metric defined in [76].

Metric	Rank	Criteria	Value
Access Vector	Local (L)	SD/USB, OBD-II, Radar/Sensor	0.395
(AV)	Adjacent (A)	CWC, BT/Wi-Fi/IR, DSRC, GPA	0.646
	Network (N)	Mobile Communication	1.000
Access Complexity	High (H)	3 or more	0.350
(AC)	Middle (M)	2	0.610
	Low (L)	1	0.710
Authentication	Multiple (M)	2 or more	0.450
(Au)	Single (S)	1	0.560
	None (N)	0	0.704

With respect to CGW, our study makes the following points clear:

(i) Regarding the attack path, the entry point via the telematics leads to a Level III threat. This was not expected before our study. Intuitively, the OBD interface might lead to this kind of threat. However, this is not the case. Some connected cars have already had updated software (and firmware) OTA by using telematics, and the number of cars updated over the air will certainly increase in the future. Attackers will be using the OTA update mechanism to reprogram ECUs to take control of the vehicle. To prevent this attack, authenticity and integrity need to be updated and implemented in CGW. Threats from telematics and attacks to CGW control functions impact safety mechanisms.

(ii) From our risk analysis result with respect to CGW, there is no significant bias among the 15 related

Table 15: IN_{3-4}: Effect Factor for CRSS: Value for Rank of Each Metric defined in [76].

#	Module name	Asset to be protected	Confidentiality			Integrity			Availability		
			None	Partial	Complete	None	Partial	Complete	None	Partial	Complete
			0.000	0.275	0.660	0.000	0.275	0.660	0.000	0.275	0.660
1	Power-train	Control function	√					√			√
		Authentication function	√					√			√
		Authentication information			√			√	√		
		Sensor information	√					√	√		
2	Chassis	Control function	√					√			√
3	Body	Control function	√				√			√	
		Control input data	√					√	√		
4	ADAS	Control function	√					√			√
		Sensor information	√				√		√		
5	CGW	Control function	√					√			√
	(Central Gateway)	Flow/Storage data	√					√	√		
6	ITS	Authentication function	√				√			√	
		Authentication information			√			√	√		
		Control information	√				√			√	
7	Telematics	Authentication function	√				√			√	
		Authentication information			√			√	√		
		Personal information			√			√	√		
		Request information to server	√				√			√	
		Vehicle status information		√			√		√		
8	Infotainment	Authentication function	√					√			√
		Authentication information			√			√	√		

Table 16: Definition of Risk Level for CRSS [26, 27].

Risk Level	Risk Value
Level III (Critical)	7.0-10.0
Level II (Warning)	4.0-6.9
Level I (Caution)	0.8-3.9

threats with respect to the interface. This implies that uniform security countermeasures are necessary, in contrast to focusing on protecting attacks via one or a few interfaces (see Table 18).

6.4. Results of Phase 3 in the RSS-CVSSv3 Case. As mentioned in Section 5.2.2, we prioritized the threats using the RSS-CVSSv3 method in this case study (see Table 20).

6.4.1. IN_{3-7}: Exploitability for RSS-CVSSv3. In Table 19, Exploitability list is used for RSS-CVSSv3. Table 19 shows the metrics and the criteria related to exploitability. These exploitability metrics indicate ease of attack from the viewpoint of the attack route similar to the AE metrics in the CRSS method. We allocate a combination of the entry point and the module to the rank of metric.

6.4.2. IN_{3-8}: Impact for RSS-CVSSv3. IN_{3-8}: Impact for RSS-CVSSv3 is similar to the EF metrics of IN_{3-4} as shown in Appendix.

6.4.3. OUT_{3-1}: Prioritized Threat List in the RSS-CVSSv3 Case. Table 20 is the prioritized thread list.

6.5. Results on Comparison of Proposed RSS-CVSSv3 with JASO CRSS. Because CVSSv3 is the evolved system of CVSSv2, the result of RSS-CVSSv3 is similar to that of CRSS. But there are some changes. For example, the number of ranks for the Attack Vector becomes four and the value of the rank "Local(L)" increases (0.395 to 0.55) in RSS-CVSSv3. As a result, the risk assessment result from OBD-II has shifted to the higher level (see Table 21).

This is the result obtained by classifying OBD-II as a higher-level network by dividing it from other physical access points such as USB, because exploiting OBD-II can affect a wider range. Since research about attacks via OBD-II has increased in recent years, we think this difference is timely. Furthermore, because the threats that ranked higher are related to CGW, the countermeasures against the CGW are also urgent (Table 20).

Table 17: OUT_{3-1}: Prioritized Threat List.

#	Where	Who	When	Why	What	(At,	Asset)	AV	AC	Au	AE	EF-C	EF-I	EF-A	EF	Risk Value
117	Mobile (Telematics)	Outsider	In regular use	Maliciously	steal information	Telematics	Authentication information	N	L	S	8.0	C	C	N	9.2	8.47
118	Mobile (Telematics)	Outsider	In regular use	Maliciously	steal information	Telematics	Personal information	N	L	S	8.0	C	C	N	9.2	8.47
72	Mobile (Telematics)	Outsider	In regular use	Maliciously	cause malfunction	CGW	Control function	N	M	S	6.8	N	C	C	9.2	7.95
6	CWC (TPMS)	Outsider	regular use	Maliciously	cause malfunction	Power-train	Control function	A	L	S	5.1	N	C	C	9.2	7.15
7	CWC (TPMS)	Outsider	regular use	Maliciously	interfere with access	Power-train	Authentication function	A	L	S	5.1	N	C	C	9.2	7.15
8	CWC (TPMS)	Outsider	regular use	Maliciously	steal information	Power-train	Authentication information	A	L	S	5.1	C	C	N	9.2	7.15
87	DSRC (ITS)	Outsider	In regular use	Maliciously	steal information	ITS	Authentication information	A	L	S	5.1	C	C	N	9.2	7.15
141	BT/Wi-Fi/IR (Infotainment)	Outsider	In regular use	Maliciously	interfere with access to	Infotainment	Authentication function	A	L	S	5.1	N	C	C	9.2	7.15
142	BT/Wi-Fi/IR (Infotainment)	Outsider	In regular use	Maliciously	steal information	Infotainment	Authentication information	A	L	C	5.1	C	C	N	9.2	7.15
68	CWC (TPMS)	Outsider	in regular use	Maliciously	cause malfunction	CGW	Control function	A	M	S	4.4	N	C	C	9.2	6.81
70	DSRC (ITS)	Outsider	In regular use	Maliciously	cause malfunction	CGW	Control function	A	M	S	4.4	N	C	C	9.2	6.81
74	BT/Wi-Fi/IR (Infotainment)	Outsider	In regular use	Maliciously	cause malfunction	CGW	Control function	A	M	S	4.4	N	C	C	9.2	6.81
63	Radar/Sensor	Outsider	In regular use	Maliciously	cause malfunction	ADAS	Control function	L	L	N	3.9	N	C	C	9.2	6.59
64	OBD-II	Outsider	In regular use	Maliciously	cause malfunction	CGW	Control function	L	L	N	3.9	N	C	C	9.2	6.59
65	OBD-II	Owner/User	In regular use	Maliciously	cause malfunction	CGW	Control function	L	L	N	3.9	N	C	C	9.2	6.59
66	OBD-II	Maintenance staff	In maintenance	Accidentally	cause malfunction	CGW	Control function	L	L	N	3.9	N	C	C	9.2	6.59

Table 18: OUT_{3-1}: Risk Analysis with respect to CGW.

#	Where	Who	When	Why	What (At, Asset)	AE	EF	Risk Value
64	OBD-II	Outsider	In regular use	Maliciously	cause malfunction (CGW, Control function)	3.9	9.2	6.6
65	OBD-II	Owner/User	In regular use	Maliciously	cause malfunction (CGW, Control function)	3.9	9.2	6.6
66	OBD-II	Maintenance staff	In maintenance	Accidentally	cause malfunction (CGW, Control function)	3.9	9.2	6.6
67	OBD-II	Maintenance staff	In maintenance	Accidentally	write wrong data (CGW, Flow/Storage data)	3.9	6.9	4.9
68	CWC (TPMS)	Outsider	In regular use	Maliciously	cause malfunction (CGW, Control function)	4.4	9.2	6.8
69	CWC (TPMS)	Outsider	In regular use	Maliciously	write wrong data (CGW, Flow/Storage data)	4.4	6.9	5.2
70	DSRC (ITS)	Outsider	In regular use	Maliciously	cause malfunction (CGW, Control function)	4.4	9.2	6.8
71	DSRC (ITS)	Outsider	In regular use	Maliciously	write wrong data (CGW, Flow/Storage data)	4.4	6.9	5.2
72	Mobile (Telematics)	Outsider	In regular use	Maliciously	cause malfunction (CGW, Control function)	6.8	9.2	7.9
73	Mobile (Telematics)	Outsider	In regular use	Maliciously	write wrong data (CGW, Flow/Storage data)	6.8	6.9	6.3
74	BT/Wi-Fi/IR (Infotainment)	Outsider	In regular use	Maliciously	cause malfunction (CGW, Control function)	4.4	9.2	6.8
75	BT/Wi-Fi/IR (Infotainment)	Outsider	In regular use	Maliciously	write wrong data (CGW, Flow/Storage data)	4.4	6.9	5.2
76	SD/USB (Infotainment)	Outsider	In regular use	Maliciously	cause malfunction (CGW, Control function)	2.7	9.2	6.0
77	SD/USB (Infotainment)	Owner/User	In regular use	Accidentally	infect with malware (CGW, Control function)	2.7	9.2	6.0
78	SD/USB (Infotainment)	Outsider	In regular use	Maliciously	write wrong data (CGW, Flow/Storage data)	2.7	6.9	4.4

Table 19: IN_{3-7}: Exploitability for RSS-CVSSv3: Value for Rank of Each Metric is defined in[5].

Metric	Rank	Criteria	Value
Access Vector	Network (N)	Mobile Communication, GPA	0.85
(AV)	Adjacent (A)	BT/Wi-Fi/IR, DSRC	0.62
	Local (L)	OBD-II, Radar/Sensor	0.55
	Physical (P)	SD/USB	0.20
Access Complexity	Low (L)	1	0.77
(AC)	High (H)	2 or more	0.44
Privilege Required	None (N)	0	0.85
(PR)	Low (L)	1	0.62 (if Scope=U)
			0.68 (if Scope=C)
	High (H)	2 or more	0.27 (if Scope=U)
			0.50 (if Scope=C)
User Interaction	None (N)		0.85
(UI)	Required (R)		0.62

Table 20: OUT_{3-1}: Prioritized Threat List in the RSS-CVSSv3 Case.

#	Where	Who	When	Why	What	(At,	Asset)	AV	AC	PR	UI	EX	Im-C	Im-I	Im-A	ISC	SC	Risk Value
117	Mobile (Telematics)	Outsider	In regular use	Maliciously	steal information	Telematics	Authentication information	N	L	L	N	2.8	H	H	N	5.2	U	8.01
118	Mobile (Telematics)	Outsider	In regular use	Maliciously	steal information	Telematics	Personal information	N	L	L	N	2.8	H	H	N	5.2	U	8.01
63	Radar/Sensor	Outsider	In regular use	Maliciously	cause malfunction	ADAS	Control function	L	L	N	N	2.5	N	H	H	5.2	U	7.69
64	OBD-II	Outsider	In regular use	Maliciously	cause malfunction	CGW	Control function	L	L	N	N	2.5	N	H	H	5.2	U	7.69
65	OBD-II	Owner/User	In regular use	Maliciously	cause malfunction	CGW	Control function	L	L	N	N	2.5	N	H	H	5.2	U	7.69
66	OBD-II	Maintenance staff	In maintenance	Accidentally	cause malfunction	CGW	Control function	L	L	N	N	2.5	N	H	H	5.2	U	7.69
6	CWC (TPMS)	Outsider	regular use	Maliciously	cause malfunction	Power-train	Control function	A	L	L	N	2.1	N	H	H	5.2	U	7.25
7	CWC (TPMS)	Outsider	regular use	Maliciously	interfere with access	Power-train	Authentication function	A	L	L	N	2.1	N	H	H	5.2	U	7.25
8	CWC (TPMS)	Outsider	regular use	Maliciously	steal information	Power-train	Authentication information	A	L	L	N	2.1	H	H	N	5.2	U	7.25
87	DSRC (ITS)	Outsider	In regular use	Maliciously	steal information	ITS	Authentication information	A	L	L	N	2.1	H	H	N	5.2	U	7.25
141	BT/Wi-Fi/IR (Infotainment)	Outsider	In regular use	Maliciously	interfere with access to	Infotainment	Authentication function	A	L	L	N	2.1	N	H	H	5.2	U	7.25
142	BT/Wi-Fi/IR (Infotainment)	Outsider	In regular use	Maliciously	steal information	Infotainment	Authentication information	A	L	L	N	2.1	H	H	N	5.2	U	7.25
72	Mobile (Telematics)	Outsider	In regular use	Maliciously	cause malfunction	CGW	Control function	N	H	L	N	1.6	N	H	H	5.2	U	6.80
14	Mobile (Telematics)	Outsider	In regular use	Maliciously	cause malfunction	Power-train	Control function	N	H	L	N	1.6	N	H	H	5.2	U	6.80
15	Mobile (Telematics)	Outsider	In regular use	Maliciously	interfere with access to	Power-train	Authentication function	N	H	L	N	1.6	N	H	H	5.2	U	6.80
16	Mobile (Telematics)	Outsider	In regular use	Maliciously	steal information	Power-train	Authentication information	N	H	L	N	1.6	H	H	N	5.2	U	6.80
31	Mobile (Telematics)	Outsider	In regular use	Maliciously	cause malfunction	Chassis	Control function	N	H	L	N	1.6	N	H	H	5.2	U	6.80
56	Mobile (Telematics)	Outsider	In regular use	Maliciously	cause malfunction	ADAS	Control function	N	H	L	N	1.6	N	H	H	5.2	U	6.80
90	Mobile (Telematics)	Outsider	In regular use	Maliciously	steal information	ITS	Authentication information	N	H	L	N	1.6	H	H	N	5.2	U	6.80
139	Mobile (Telematics)	Outsider	In regular use	Maliciously	interfere with access to	Infotainment	Authentication function	N	H	L	N	1.6	N	H	H	5.2	U	6.80
140	Mobile (Telematics)	Outsider	In regular use	Maliciously	steal information	Infotainment	Authentication information	N	H	L	N	1.6	H	H	N	5.2	U	6.80

Table 21: Comparison of Results: CRSS vs RSS-CVSSv3.

			CRSS		RSS-CVSSv3	
#	Where	At	Risk Value	Rank	Risk Value	Rank
64	OBD-II	CGW	6.59	13	7.69	2
65	OBD-II	CGW	6.59	13	7.69	2
66	OBD-II	CGW	6.59	13	7.69	2

6.6. Results of Phase 3 in the Q-RSMA Case. As mentioned in Section 5.3, we prioritized the threats using the Q-RSMA method in this case study.

Q-RSMA uses the same Occurrence Possibility list used for RSMA; $IN_{3-9} = IN_{3-5}$

Table 22 is the prioritized thread list. From the perspectives of "Who," "When," and "Why," assigned values are always the same. The "Who" is 'Outsider,' the "When" is 'In regular use,' and "Why" is 'Maliciously,' respectively, in all the 14 high risk threats. Hence, these corresponding three columns are omitted from this table.

Although the top two show the same results as in Table 17, 8 threats are newly replaced. With their tendency to change priority, "Opportunity (O)" and "Device (D)" metrics seem to emphasize the risk value at Q-RSMA. These threats are emphasized because there is enough opportunity to attack and the attack is easy to carry out by remodeling commercial devices.

In addition, depending on the features of Q-RSMA, this method uses multiplication of "Occurrence Possibility and Effect Factor," and the magnitude of the Effect Factor tends to influence the result significantly. For example, #141 is the fourth highest value tied in Table 17 but is the 141st in the Q-RSMA result (Value=2.30), because the asset "The Authentication Function of Infotainment" is not considered important (property/corporate value = Small) and is one-third the value compared with High (Table 23).

As mentioned in Table 23, the differences are seen as the method evaluated from the attack scenario or as a quantification method by multiplication compared with CRSS.

6.7. Safety-Related Threats Included the Result of Phase 3 in Each Case. It is interesting that the threats as an output of Phase 3 include some safety-related threats.

Both the threat on the Power-Train ECU (access) and the threat on the Chassis ECU (brake and steering) can be viewed as safety-related threats. In this case study, Threat #6, #14, and #31 are the safety-related threats.

For comparison purposes, we now examine three safety-related threats shown in Table 24, for which the risk values and the prioritized rank are associated. Although there are some differences in risk value between CRSS and RSS-CVSSv3, they are simply rank fluctuations because the classification change of Access Complexity (AC) has been made to CVSSv2 on which CRSS is based. Consequently, we consider that there is no fundamental difference in risk assessment between CRSS and RSS-CVSSv3 regarding these three safety-related threats.

For each of the three methods that we used in this study, the Impact metrics reflect how important the asset is and the Exploitability metrics reflect the ease and the technical means including the route in which the attack can be mounted from a remote place at any time (by the devices easy to purchase in the Q-RSMA case). Hence, their scores belong to a higher rank in the prioritized threat list in each case. As described above, our proposed methods identified the safety-related threats in this case study.

7. Case Study 2

In this section, we describe the components of the intradrone network formed by the interconnection of the multiple modules in the drone.

There are two reasons why we add this case study:

(i) **To verify whether our proposed method can be applied to other systems**, that is, not only limited to the automobile industry. Precisely, we are going to confirm whether the anticipated threats are reasonably quantified by our proposed method.

(ii) **To visualize the threats influenced by new features**, which are likely to be applied to automated driving technology. The drone has an autonomous control function in addition to a remote operation function, and we attempt to investigate what threats will be regarded as more important in the future, such as attacks on logs of the sensor information and position information in this case study.

7.1. Results of Phases 1 and 2

7.1.1. OUT_{1-1}: TOE Model. There is prior research on attacks on drones where a threat model is typically established before the actual attacks are described. In [8], the threat model considers an attacker that is within Wi-Fi range of the drone but otherwise has no other direct physical access to the device. In their drone model, drones have Wi-Fi access points for their devices to be connected. According to the authors, it is very common in IoT systems that embedded devices with a Wi-Fi access point are used to communicate with an application in a smart phone.

The sensor regarding altitude, called an optical flow sensor, is used for the drone to attempt to keep the altitude constant. In [78], the authors demonstrate the effectiveness of their optical flow sensor input spoofing attack against certain drones.

Table 22: OUT_{3-1}: Prioritized Threat List in the Q-RSMA Case.

#	Where	What	(At,	Asset)	Type	Effect Factor	T	E	TOE	O	D	Risk Value
117	Mobile (Telematics)	steal information	Telematics	Authentication information	Privacy	L	P	N	L	A	OS	9.30
118	Mobile (Telematics)	steal information	Telematics	Personal information	Privacy	L	P	N	L	A	OS	9.30
63	Radar/Sensor	cause malfunction	ADAS	Control function	Property/Value	L	P	N	L	L	OS	8.10
90	Mobile (Telematics)	steal information	ITS	Authentication information	Privacy	L	P	N	C	A	OS	8.10
140	Mobile (Telematics)	steal information	Infotainment	Authentication information	Privacy	L	P	N	C	A	OS	8.10
145	SD/USB (Infotainment)	steal information	Infotainment	Authentication information	Privacy	L	P	N	L	L	OS	8.10
6	CWC(TPMS)	cause malfunction	Power-train	Control function	Safety	L	P	P	L	L	OS	7.20
14	Mobile (Telematics)	cause malfunction	Power-train	Control function	Safety	L	P	P	C	A	OS	7.20
31	Mobile (Telematics)	cause malfunction	Chassis	Control function	Safety	L	P	P	C	A	OS	7.20
56	Mobile (Telematics)	cause malfunction	ADAS	Control function	Property/Value	L	P	P	C	A	OS	7.20
58	BT/Wi-Fi/IR (Infotainment)	cause malfunction	ADAS	Control function	Property/Value	L	P	P	L	L	OS	7.20
72	Mobile (Telematics)	cause malfunction	CGW	Control function	Property/Value	L	P	P	C	A	OS	7.20
73	Mobile (Telematics)	write wrong data	CGW	Flow/Storage data	Property/Value	L	P	P	C	A	OS	7.20
91	Mobile (Telematics)	write wrong data	ITS	Control information	Property/Value	L	P	P	C	A	OS	7.20

Table 23: Risk Value Comparison at Threat #141.

	CRSS							Q-RSMA							
#	AV	AC	Au	EF-C	EF-I	EF-A	Value	Type	Effect Factor	T	E	TOE	O	D	Value
141	A	L	S	C	C	N	7.15	Property/Value	S	P	N	C	L	OS	2.30

Table 24: Safety-related Threats in Case Study 1.

		What		CRSS		RSS-CVSSv3		Q-RSMA	
#	Where	At	Asset	Risk Value	Rank	Risk Value	Rank	Risk Value	Rank
6	CWC (TPMS)	Power-train	Control function	7.15	4	7.25	7	7.20	7
14	Mobile (Telematics)	Power-train	Control function	6.58	17	6.80	13	7.20	7
31	Mobile (Telematics)	Chassis	Control function	6.58	17	6.80	13	7.20	7

On the other hand, in our TOE model, in addition to a wireless communication interface, there are more than one entry point. We also refer to recent research on drones [79, 80].

In Figure 5, TOE is defined as the area surrounded by the dotted line, which realizes communication control as a connection interface to the outside of the drone.

7.1.2. OUT_{1-2}: Module Function List. The Module Function List in TOE OUT_{1-2} is shown and the main security perspectives are Confidentiality (C), Integrity (I), and/or Availability (A) in order to provide security requirements as the goal of security design.

In a drone, sensors such as cameras, altitude sensors, geomagnetic sensors, and navigational systems are typically present. Drones use several kinds of sensors to fly safely. Many of them have Global Navigational Satellite Systems (GNSS) such as GPS receivers. A downward-facing optical flow camera is used to stabilize the drone.

The drone needs to use its sensors to react to its environment in a timely manner. Therefore, for instance, the cameras record video image and send this data to the drone's navigation system to interpret the video image and trigger commands for navigation.

The battery is critical to flying safely. Lithium Polymer (Li-Po) batteries are known as good technology in terms of energy density, power density, and lifetime use.

7.1.3. OUT_{1-3}: Lifecycle List. Table 26 shows the lifecycle list related to our drone system. This is similar to the case of the connected-car system except that the user performs the maintenance.

7.1.4. OUT_{2-1}: Assumption List. Table 27 shows a description of assumptions. Compared to Case Study 1 (connected-car), the drone is conscious of autonomous movements, and attacks on sensors and the integrity of information necessary for flight are emphasized.

7.2. Results of Phase 3 in the CRSS Case. By applying TP 15002 (CRSS) and our method, our study identified 138 threats on our drone systems. In concrete, we applied our "Asset Container" method to the drone TOE as follows:

(i) Figure 5 showed that the TOE had 7 "Where"(entry points).

(ii) Table 25 showed that it had 26 combinations of "At" and "Asset."

(iii) Therefore, it was confirmed that the TOE could be subject to at least $182 = 7 \times 26$ kinds of attacks.

(iv) $96 = 4 \times 24$ kinds of attacks were excluded because the attack from 4 entry points could only attack to each first module (see A.Attack_to_Sensor in Table 27).

(v) 52 kinds of attacks were added by applying STRIDE method which added the variants of "Who," "Why," and "What."

(vi) Finally $138 = 182 - 96 + 52$ threats were listed.

Table 28 shows a part of the prioritized threat list for our drone system (OUT_{3-1}) where there are 15 threats with risk value, ranging from 7.51 to 9.43, categorized as level III, the most important class of threats. From the perspectives of "Who" and "Why," the assigned values are always the same. The "Who" is 'Outsider,' and the "Why" is 'Maliciously,' respectively, in all the 15 high risk threats. Hence these corresponding two columns are omitted from this table.

In these threats, the entry points for the attackers that come to TOE are mainly GPA and wireless communication, the remote network. Threat #26 corresponds to a lock-out attack [8] where the attacker prevents the legitimate owner of the device from connecting to it.

The threat to the electromagnetic wave and supersonic wave is ranked in, too. This fact indicates the importance of sensors. Also, the attacks targeting information necessary for autonomous flight are also ranked in the top rank. These are the features of our drone system.

7.3. On the Relevance of This Study to Real-World Drone Security. We believe that this study is of practical relevance to real-world Drone Security, so that Threat #27 and Threat

Table 25: OUT_{1-2}: Module Function List for Our Drone System.

#	Module name	Function	Assets to be protected	C	I	A
1	CPU	Control function module for flight:	Firmware	√	√	
	(Flight Controller)	(1) Control motors using the Control command from the	Configuration information		√	
		Receiver and the Configuration information,	Control function		√	√
		(2) Keep altitude from the information of the Altitude Sensor,	Control command	√	√	
		(3) Keep direction from the information of the Geomagnetic Sensor,	Data handling function		√	√
		(4) Send the video image of the Camera to the Storage and the	Status information		√	
		Receiver,	Location information	√	√	
		(5) Edit the Status information from the data of the Sensors, the	Position information	√	√	
		GPS, and the Battery, and send it to the Receiver,	Battery information		√	
		(6) Flies back automatically if communication is interrupted.	Video image	√	√	
2	Receiver	Communication functions to access with the Smart Phone:	Authentication function		√	√
	(Network Interface)	(i) Receive the Control Command,	Authentication information	√	√	
		(ii) Send the Status information and the Video image,	Communication function		√	√
		(iii) Use wireless communication with authentication	Control command	√	√	
			Status information		√	
			Video image	√	√	
3	GPS	Receive the signal from GPS, and calculate the Location	Receive function		√	√
		information	Location information	√	√	
4	Camera	Record the Video image	Recording function		√	√
			Video image	√	√	
5	Position Sensor	Calculate the Altitude, Attitude, and Direction of the Drone,	Sensor function		√	√
	(Altitude, Attitude, Direction)		Position information	√	√	
6	Storage	Store the Video image to the SD-card	Storage function		√	√
			Video image	√	√	
7	Li-Po Battery	Supply the power to the Drone and charge the power from the	Charging function		√	√
		external connector	Battery information		√	

Table 26: OUT_{1-3}: Life Cycle List for Our Drone System.

Phase	Subphase	Overview	Participants
Plan	-	This is the phase in which the planner and the developers of the manufacturer plan their product. They formulate their product's concept, for example, the scope of user, how to use the services of their product in the field. They also summarize and determine the required specifications of the product which includes functional and nonfunctional items. They evaluate the entire cost of the product throughout its entire life cycle at this phase, and decide on both the safety level and "the security level" of the product in addition to the safety of the product. It is important for the requirement definitions to include the security requirements and not to include the vulnerabilities.	Supplier staff
Development	Product design	This is the phase in which the developers of the drone's manufacture and the parts maker design the hardware and the software based on the requirement definitions of the plan phase, where they implement and manufacture the product. It is necessary at this phase that "the requirement definitions are correctly implemented in the product," "the vulnerabilities shall not be included in the product," and "the vulnerabilities, if included must be detected before the shipment of the product."	Supplier staff
	Manufacturing		Supplier staff
Operation	Shipping	This is the phase in which the user buys the drone through the agency of the manufacture (or rents the drone through a rental lease company), and flies it. For example, the user operates the drone, collects various information such as video images, Also when a user buys the second-hand drone through a second-hand broker, it is necessary for the broker to consider both what should be kept ongoing (e.g., the latest security patch of the firmware) and what must be erased in advance (e.g., the personal information of the previous owner) of the accumulated information. In the case of a rental lease, it is also necessary for the user to check if all recorded video images and private information are erased. And it is also necessary for the agency staff to let the user know of the vulnerabilities if some are found after the shipment.	Carrier staff
	Registration		Agency staff Second-hand broker Rental lease company's staff
	Regular use, operation		User Outsider
	Maintenance		Agency staff User
Disposal	-	This is the phase in which the user sells or discards the product to replace it or if it breaks down. There are two cases to let go the product, one is the case that the owner sells the product to other user through the second-hand broker, the other is the case that the user makes it scrapped. Anyway, it is necessary some procedures (in-person inspection, issuing evidence, etc.) that the user can confirm that all confidential data, for example, the private information and the authentication information, are erased.	User Dismantler Second-hand broker

Table 27: OUT_{2-1}: Assumption List for Our Drone System.

Type	Assumption	Overview
System	A.Semi_Self_Control	The TOE has an attitude and altitude control, so the integrity of all information is necessary to assist the user.
Module	A.CPU_Not_Disclosed_Firmware	The firmware is not disclosed and difficult to retrieve for analysis.
	A.CPU_Not_Disclosed_Configuration	The configuration information is not disclosed and difficult to retrieve for analysis.
	A.Attack_to_Sensor	The attacks to the sensor modules (Camera, Position Sensor, GPS, and Battery) from their entry points (Optical wave, Electromagnetic wave, GPA, and Physical) only influence themselves (single-stage attack only).

#33 in Table 28 are related to the vulnerabilities in commercial drones [8].

In 2016, attacks via optical flow were reported in [78], where the authors demonstrate the effectiveness of this sensor input by mounting spoofing attacks on certain drones. From this point of view, additional relevance can be illustrated by our risk assessment comparison on Threat #6 that exploited sensor input shown in Table 29. It is interesting to us that, for this threat, our proposed RSS-CVSSv3 suggests a higher risk level, namely, three, than the other two methods.

8. Discussion

8.1. Number of Metrics Necessary for Risk Assessment Methods to Work Properly. It is worth discussing how many metrics a risk assessment method needs for the risk scores to be moderately distributed. Our intuition is that the number of these metrics may be about 5 or 6 based on what our case studies suggest. It seems to us that, if these metrics are classified in an appropriate manner, the resulting risk assessment method including the risk ranking would work properly.

8.2. Evaluation of Assets in the Asset Container Method. However, it does not appear that the risk scores are effectively distributed, in spite of the number of metrics. This ineffectiveness can be illustrated by our case-study example showing that there are six threats whose scores are the same, that is, 7.15 in CRSS, there are nine threats whose scores are the same, 6.80 in RSS-CVSSv3, and there are eight threats whose scores are the same 7.30 in Q-RSMA. One reason why several different threats tend to group in the same risk value could be due to the rough way of classifying the impact related to the asset.

Also, in Q-RSMA, where there are only four values of Effect Factor, the classification of CRSS and RSS-CVSSv3 are also rough, which can be explained because the Impact values are only a few at best, even though the asset values can be selected out of three values for each of the three metrics (Confidentiality, Integrity, and Availability).

It is possible that the other metrics concerning the attack route and method make a difference in the risk score, which does not mean that we can neglect to strictly evaluate the value of each asset.

Our proposed method of an "Asset Container" allows the system developer implementing JASO TP15002 to identify the threats in a systematic manner, without considering how the attack is mounted and what the attack means is. Although it is difficult to cover all attack variations, that is, to imagine the attacker's potential, we can grasp all the assets and the routes to them. We think that a combination method is preferable, as it ensures comprehensiveness of threats with the "Asset Container" method, prioritizes them in risk assessment, and then attempts to cover means of them by the STRIDE method and so on. Therefore, it is important for us to classify and prioritize the assets at first. For the sake of the developers, different assets corresponding to similar values in the risk assessment phase should be avoided.

8.3. Evaluation of Routes in the Asset Container Method. We have confirmed that the other metrics concerning the attack routes, that is, the metrics of Attack Ease in CRSS and Exploitability in RSS-CVSSv3, were effective in identifying the important threats. Specifically, by comparing Tables 17 and 20, we notice that the rank of threats via OBD-II [31], which have received much attention in recent years, has become higher in RSS-CVSSv3 because the degree of attention to the metric has been changed.

8.4. On the Use of Different Methods for Threat Identification. In this paper, we mainly assume the same method threat identification by means of 5W is used for the existing methods in TP15002 and the proposed methods for risk assessment. It appears to us that the 5W method is suitable for these risk assessment methods in terms of risk score computation. On the other hand, it would be interesting whether we could find desired tradeoffs between time spent in the security design task and the granularity of the result. This could give the designers a variety of choices. We speculate that there are certain designers who would need a rough result with limited time.

9. Limitations

The scope of this research is limited in the sense that we only discuss the results up to Phase 3 in JASO TP15002. Our main focus is on the trend of how the risk values are taken and distributed through the comparison result of risk assessment methods. We are currently considering whether an efficient definition of the security objectives can be realized by efficiently distributing risk values at Phase 4:

Table 28: OUT_{3-1}: Prioritized Threat List for Our Drone System.

#	When	Where	What	(At,	Asset)	AV	AC	Au	AE	EF-C	EF-I	EF-A	EF	Risk Value
1	Operation	GPA	Malfunction	GPS	Receive function	N	L	N	10.0	N	C	C	9.2	9.43
2	Operation	GPA	Tampering	GPS	Location information	N	L	N	10.0	C	C	N	9.2	9.43
26	Operation	Wireless	Malfunction	Receiver	Authentication function	N	L	S	8.0	N	C	C	9.2	8.47
27	Operation	Wireless	Leakage	Receiver	Authentication information	N	L	S	8.0	C	C	N	9.2	8.47
28	Operation	Wireless	Malfunction	Receiver	Communication function	N	L	S	8.0	N	C	C	9.2	8.47
10	Operation	Wireless	Tampering	CPU	Firmware	N	M	S	6.8	C	C	N	9.2	7.95
11	Purchase	Wireless	Leakage	CPU	Firmware	N	M	S	6.8	C	C	N	9.2	7.95
13	Operation	Wireless	Malfunction	CPU	Control function	N	M	S	6.8	N	C	C	9.2	7.95
15	Operation	Wireless	Malfunction	CPU	Data handling function	N	M	S	6.8	N	C	C	9.2	7.95
18	Operation	Wireless	Tampering	CPU	Location information	N	M	S	6.8	C	C	N	9.2	7.95
19	Operation	Wireless	Leakage	CPU	Location information	N	M	S	6.8	C	C	N	9.2	7.95
5	Operation	Electromagnetic wave/ Supersonic wave	Malfunction	Position Sensor	Sensor function	A	L	N	6.5	N	C	C	9.2	7.77
29	Operation	Wireless	Tampering	Receiver	Control command	N	L	S	8.0	P	C	N	7.8	7.51
32	Operation	Wireless	Tampering	Receiver	Video image	N	L	S	8.0	C	P	N	7.8	7.51
33	Operation	Wireless	Leakage	Receiver	Video image	N	L	S	8.0	C	P	N	7.8	7.51

Table 29: Risk Score Comparison for Threat #6 Exploiting the Sensor Input.

#	Who	When	Why	Where	What	(At	Asset)	CRSS Score, Level	RSS-CVSSv3 Score, Level	Q-RSMA Score, Level
6	Outsider	Operation	Maliciously	Electromagnetic wave/ Supersonic wave	Tampering	Position Sensor	Position information	6.81 Level II	7.05 Level III	6.00 Level II

define security objectives or phase 5: security requirement selection in JASO TP15002. Our discussions in this paper regarding by which operation the risk score is derived from the multiple metrics, addition or multiplication, are limited. The only point we make clear is that the same technical paper TP15002 specifies the different approaches, namely, CRSS and RSMA. CRSS uses addition while RSMA uses multiplication.

10. Conclusion

We have investigated a promising security JASO TP15002 guideline. We have identified the difficulties in carrying out the actual design tasks based on JASO TP15002 and we have proposed a systematic way for implementing the 3-phase risk assessment methodology in the guideline.

For the systematic risk assessment, we introduced the idea of an asset container and proposed extending CRSS to a novel Risk Scoring System (RSS)s, RSS-CVSSv3, by appropriately replacing the CVSSv2 vulnerability scoring system on which CRSS is based with CVSSv3. We performed a comparative study on CRSS, RSMA, and RSS-CVSSv3 for multiple use cases such as a CGW and a drone to examine the efficiency and usefulness of our methods. For this comparison purpose, we devised an interesting approach of refinement of RSMA to the obstacles in comparing CRSS and RSMA. As a result, we have provided some interesting data so that the system developers can have a better understanding on original and proposed risk assessment methods in a comparative manner. We expect that our research will offer an information basis on which the system designers and developers can choose suitable security risk assessments for the future system designers of connected vehicles.

Appendix

A. Inputs for Case Study 1

A.1. $\boldsymbol{IN}_{3-8}$*: Impact for RSS-CVSSv3.* Table 30 shows the metrics and the criteria related to impact of asset if it will be attacked. This impact metric indicates importance of asset from the viewpoint of the security CIA and the seriousness when the attack succeeds, the same as the Effect Factor (EF) at CRSS method.

B. Inputs for Case Study 2

B.1. $\boldsymbol{IN}_{3-3}$*: Attack Ease.* See Table 31.

B.2. $\boldsymbol{IN}_{3-4}$*: Effect Factor for CRSS.* See Table 32.

Abbreviated Terms

ADAS: Advanced Driver-Assistance Systems
AE: Attack Ease
ASIL: Automotive Safety Integrity Level
AT: Attack Tree
BT: Bluetooth (Communication)
CAL: Cybersecurity Assurance Level
CAN: Controller Area Network
CEM: Common Evaluation Methodology for Information Technology Security Evaluation
CC: Common Criteria
CERT: Computer Emergency Response Team
CGW: Central Gateway
CIA: Confidentiality, Integrity, and Availability
CRSS: CVSS Based Risk Scoring System
CVSS: Common Vulnerability Scoring System
CWC: Close-Range Wireless Communication
DFD: Data Flow Diagram
DSRC: Dedicated Short Range Communications
ECU: Electronic Control Unit
EF: Effect Factor
FT: Fault Tree
FTP: File Transfer Protocol
GPA: GPS Antenna
GPS: Global Positioning System
JSAE: Society of Automotive Engineers of Japan, Inc.
JASO: Japanese Automotive Standards Organization
IR: Infrared (Communication)
ITS: Intelligent Transport Systems
OBD-II: On-Board Diagnosis second generation
OTA: Over-The-Air
Q-RSMA: Quantifiable-RSMA
RSMA: Risk Scoring Methodology for Automotive system
RSS: Risk Scoring System
STRIDE: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege
TOE: Target of Evaluation
TPMS: Tire Pressure Monitoring System
UAV: Unmanned Aerial Vehicle
VMG: Vehicle Mobile Gateway.

Table 30: IN_{3-8}: Impact for RSS-CVSSv3.

#	Module name	Asset to be protected	Confidentiality (Im-C)			Integrity (Im-I)			Availability (Im-A)		
			None	Low	High	None	Low	High	None	Low	High
			0.00	0.22	0.56	0.00	0.22	0.56	0.00	0.22	0.56
1	Power-train	Control function	√					√			√
		Authentication function	√					√			√
		Authentication information			√			√	√		
		Sensor information	√					√	√		
2	Chassis	Control function	√					√			√
3	Body	Control function	√				√			√	
		Control input data	√					√	√		
4	ADAS	Control function	√					√			√
		Sensor information	√				√		√		
5	CGW (Central Gateway)	Control function	√					√			√
		Flow/Storage data	√					√	√		
6	ITS	Authentication function	√				√			√	
		Authentication information			√			√	√		
		Control information	√				√			√	
7	Telematics	Authentication function	√				√			√	
		Authentication information			√			√	√		
		Personal information			√			√	√		
		Request information to server	√				√			√	
		Vehicle status information		√			√		√		
8	Infotainment	Authentication function	√					√			√
		Authentication information			√			√	√		

Data Availability

The data used to support the findings of this study are included within the article.

Disclosure

This paper is an extended version of [1], which appeared in the proceedings of the DECSoS 2017 Workshop.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and Edit Science, Inc. (http://www.editscience.com/) for English language editing.

Table 31: $IN_{3-3-CRSS}$: Attack Ease for Our Drone System.

Metric	Classification	Rank	Criteria	Value
Access Vector (AV):	Distance from threat	Local (L)	Serial, USB, Physical connector	0.395
		Adjacent (A)	Optical wave, Electromagnetic wave, Supersonic wave	0.646
		Network (N)	Wi-Fi, GPA	1.000
Access Complexity (AC):	Number of penetrations	High (H)	3 or more	0.350
		Medium (M)	2	0.610
		Low (L)	1	0.710
Authentication (Au):	Number of authentication	Multiple (M)	2 or more	0.450
		Single (S)	1	0.560
		None (N)	0	0.704

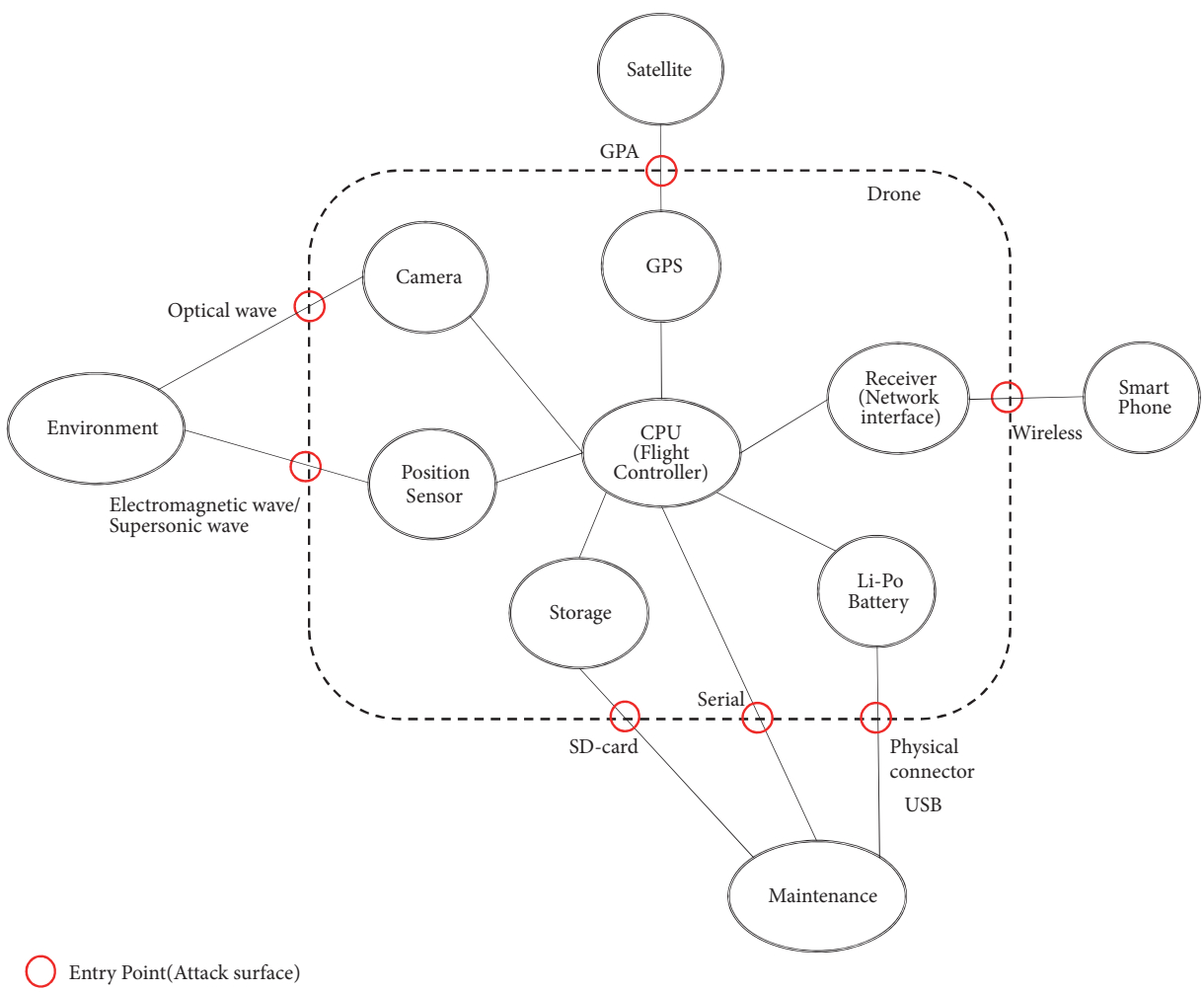


Figure 5: OUT_{1-1}: TOE Model for Our Drone System.

Table 32: $IN_{3-4-CRSS}$: Effect Factor for Our Drone System.

#	Module name	Assets to be protected	Confidentiality			Integrity			Availability		
			None 0.000	Partial 0.275	Complete 0.660	None 0.000	Partial 0.275	Complete 0.660	None 0.000	Partial 0.275	Complete 0.660
1	CPU (Flight Controller)	Firmware			√			√	√		
		Configuration information	√					√	√		
		Control function	√					√			√
		Control command		√				√	√		
		Data handling function	√					√			√
		Status information	√					√	√		
		Location information			√			√	√		
		Position information		√				√	√		
		Battery information	√					√	√		
		Video image			√		√		√		
2	Receiver (Network Interface)	Authentication function	√					√			√
		Authentication information			√			√	√		
		Communication function	√					√			√
		Control command		√				√	√		
		Status information	√					√	√		
		Video image			√		√		√		
3	GPS	Receive function	√					√			√
		Location information			√			√	√		
4	Camera	Recording function	√				√			√	
		Video image			√		√		√		
5	Position Sensor (Altitude, Attitude, Direction)	Sensor function	√					√			√
		Position information		√				√	√		
6	Storage	Storage function	√				√			√	
		Video image			√		√		√		
7	Li-Po Battery	Charging function	√				√			√	
		Battery information	√					√	√		

References

[1] Y. Kawanishi, H. Nishihara, D. Souma, and H. Yoshida, "Detailed analysis of security evaluation of automotive systems based on JASO TP15002," in *Proceedings of the SAFECOMP Workshops*, pp. 211–224, 2017.

[2] K. Koscher, A. Czeskis, F. Roesner et al., "Experimental security analysis of a modern automobile," in *Proceedings of the 31st IEEE Symposium on Security and Privacy*, pp. 447–462, May 2010.

[3] C. Miller and C. Valasek, *Adventures in Automotive Networks and Control Units*, vol. 21, DEFCON, 2013.

[4] NIST, "National vulnerability database," Tech. Rep. CVE-2016-9337, 2016, https://nvd.nist.gov/vuln/detail/CVE-2016-9337.

[5] ITU, *ITU-T X.1521: Cybersecurity information exchange, Vulnerability/state exchange, Common vulnerability scoring system*, 2016.

[6] X. Li, Y. Yu, G. Sun, and K. Chen, "Connected Vehicles' Security from the Perspective of the In-Vehicle Network," *IEEE Network*, vol. 32, no. 3, pp. 58–63, 2018.

[7] J. Liu, S. Zhang, W. Sun, and Y. Shi, "In-vehicle network attacks and countermeasures: Challenges and future directions," *IEEE Network*, vol. 31, no. 5, pp. 50–58, 2017.

[8] J. Valente and A. A. Cárdenas, "Understanding security threats in consumer drones through the lens of the discovery quadcopter family," in *Proceedings of the 2017 Workshop on Internet of Things Security and Privacy (IoT S&P)*, P. Liu, Y. Zhang, T. Benson, and S. Sundaresan, Eds., pp. 31–36, 2017.

[9] J. A. Yochim, *The Vulnerabilities of Unmanned Aircraft System Common Data Links to Electronic Attack*, U.S. Army Command and General Staff College, Fort Leavenworth, Kan, USA, 2010.

[10] A. Kim, B. Wampler, J. Goppert, I. Hwang, and H. Aldridge, "Cyber attack vulnerabilities analysis for unmanned aerial vehicles," in *Proceedings of the Infotech@Aerospace Conferences*, pp. 1–30, 2012.

[11] K. Hartmann and S. Christoph, "The Vulnerability of UAVs to Cyber Attacks - An Approach to the Risk Assessment," in *Proceedings of the 5th International Conference in Cyber Conflict (CyCon)*, pp. 1–23, 2013.

[12] C. Lin, D. He, N. Kumar, K.-K. R. Choo, A. Vinel, and X. Huang, "Security and Privacy for the Internet of Drones: Challenges and Solutions," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 64–69, 2018.

[13] I. Güvenç, F. Koohifar, S. Singh, M. L. Sichitiu, and D. Matolak, "Detection, Tracking, and Interdiction for Amateur Drones," *IEEE Communications Magazine*, vol. 56, no. 4, pp. 75–81, 2018.

[14] S. Fenz and A. Ekelhart, "Verification, validation, and evaluation in information security risk management," *IEEE Security & Privacy*, vol. 9, no. 2, pp. 58–65, 2011.

[15] G. Hatzivasilis, I. Papaefstathiou, and C. Manifavas, "Software security, privacy, and dependability: Metrics and measurement," *IEEE Software*, vol. 33, no. 4, pp. 46–54, 2016.

[16] M. S. Saleh and A. Alfantookh, "A new comprehensive framework for enterprise information security risk management," *Applied Computing and Informatics*, vol. 9, no. 2, pp. 107–118, 2011.

[17] K. Bandyopadhyay, P. P. Mykytyn, and K. Mykytyn, "A framework for integrated risk management in information technology," *Management Decision*, vol. 37, no. 5, pp. 437–445, 1999.

[18] R. K. Rainer Jr., C. A. Snyder, H. H. Carr, and R. E. X. K. Rainer, "Risk analysis for information technology," *Journal of Management Information Systems*, vol. 8, no. 1, pp. 129–147, 1991.

[19] N. Kolokotronis, C. Margaritis, P. Papadopulou, P. Kanellis, and D. Martakos, "An integrated approach for securing electronic transactions over the Web," *Benchmarking*, vol. 9, no. 2, pp. 166–181, 2002.

[20] P. Shedden, R. Scheepers, W. Smith, and A. Ahmad, "Incorporating a knowledge perspective into security risk assessments," *Journal of Information and Knowledge Management Systems*, vol. 41, no. 2, pp. 152–166, 2011.

[21] R. Bernard, "Information Lifecycle Security Risk Assessment: A tool for closing security gaps," *Computers & Security*, vol. 26, no. 1, pp. 26–30, 2007.

[22] R. A. Botha and J. H. P. Eloff, "Access control in document-centric workflow systems - An agent-based approach," *Computers & Security*, vol. 20, no. 6, pp. 525–532, 2001.

[23] P. Shamala, R. Ahmad, and M. Yusoff, "A conceptual framework of info structure for information security risk assessment (ISRA)," *Journal of Information Security and Applications*, vol. 18, no. 1, pp. 45–52, 2013.

[24] ISO/IEC, *ISO/IEC 15408: Information technology – Security techniques – Evaluation criteria for IT security*, 2009.

[25] SAE, *SAE J3061, Cybersecurity Guidebook for Cyber-Physical Vehicle Systems*, 2016.

[26] JASO, *TP15002: Guideline for Automotive Information Security Analysis*, 2015.

[27] JASO, *TP15002: Guideline concerning Automotive Information Security*, 2015.

[28] ITU, *ITU-T X.1373: Secure Software Update Capability for Intelligent Transportation System Communication Devices*, 2012.

[29] World Forum for Harmonization of Vehicle Regulations (WP.29), UN Task Force on Cyber security, and OTA issues (CS/OTA), *Draft Recommendation on 'Secure software update capability for intelligent transportation system communication devices,'*, CS/OTA, 2016, https://www2.unece.org/wiki/pages/viewpage.action?pageId=40829523.

[30] C. Miller and C. Valasek, *Remote Exploitation of an Unaltered Passenger Vehicle*, Black Hat, 2015.

[31] K.-T. Cho and K. G. Shin, "Error handling of in-vehicle networks makes them vulnerable," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS'16)*, pp. 1044–1055, October 2016.

[32] R. Kameoka, T. Kubota, M. Shiozaki, M. Shirahata, R. Kurachi, and T. Fujino, "Bus-Off Attack against CAN ECU using Stuff Error injection from Raspberry Pi," in *Proceedings of the Symposium on Cryptography and Information Security (SCIS)*, Niigata, Japan, 2017 (Japanese).

[33] A. Palanca, E. Evenchick, F. Maggi, and S. Zanero, "A stealth, selective, link-layer denial-of-service attack against automotive networks," in *Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, Springer, Cham, Switzerland, 2017.

[34] S. Checkoway, D. McCoy, B. Kantor et al., "Comprehensive experimental analyses of automotive attack surfaces," in *Proceedings of the 20th USENIX conference on Security*, 2011.

[35] Q. Luo and J. Liu, "Wireless telematics systems in emerging intelligent and connected vehicles: Threats and solutions," *IEEE Wireless Communications Magazine*, pp. 1–7, 2018.

[36] NIST, "National Vulnerability Database," Tech. Rep. CVE-2018-9318, 2018, https://nvd.nist.gov/vuln/detail/CVE-2018-9318.

[37] S. Tillich and M. Wójcik, "Security analysis of an open car immobilizer protocol stack," in *Proceedings of the Trusted Systems, 4th International Conference, INTRUST 2012*, C. J. Mitchell and

A. Tomlinson, Eds., vol. 7711 of *Lecture Notes in Computer Science*, pp. 83–94, Springer, London, UK, 2012.

[38] R. Verdult and F. D. Garcia, "Cryptanalysis of the Megamos Crypto Automotive Immobilizer," *USENIX*, vol. 40, no. 6, pp. 17–22, 2015.

[39] I. Rouf, R. Miller, H. Mustafa et al., "Security and privacy vulnerabilities of in-car wireless networks: a tire pressure monitoring system case study," in *Proceedings of the 19th USENIX Conference on Security*, p. 21, 2010.

[40] Vulnerability Notes Database, "Vulnerability Note VU#334207 - DBPOWER U818A WIFI quadcopter drone allows full file-system permissions to anonymous FTP," 2017, https://www.kb.cert.org/vuls/id/334207.

[41] Common Weakness Enumeration, "CWE-276 - incorrect default permissions," 2017, https://cwe.mitre.org/data/definitions/276.html.

[42] City of Seattle, "City Attorney's Office prevails in drone case," 2017, https://news.seattle.gov/2017/01/13/city-attorneys-office-prevails.

[43] R. A. Caralli, J. F. Stevens, L. R. Young, and W. R. Wilson, "Introducing OCTAVE allegro: improving the information security risk assessment process," Tech. Rep. CMU/SEI-2007-TR-012, 2007.

[44] A. R. Ruddle, D. Ward, B. Weyl et al., "Security requirements for automotive on-board networks based on dark-side scenarios," *E-Safety Vehicle Intrusion Protected Applications (EVITA)*, 2009.

[45] A. Barber, "Status of work in process on ISO/SAE 21434 automotive cybersecurity standard," 2018, https://www.sans.org/summit-archives/file/summit-archive-1525889601.pdf.

[46] K. V. Prasad, T. J. Giuli, and D. Watson, "The Case for Modeling Security, Privacy, Usability and Reliability (SPUR) in Automotive Software," Model-Driven Development of Reliable Automotive Services LNCS 4922, 2008.

[47] B. Dobbing and S. Lautieri, "SafSec Methodology: Standard," *Praxis High Integrity Systems*, no. 3.1, S.P1199.50.2, 2006.

[48] D. G. Firesmith, "Common concepts underlying safety, security, and survivability engineering," Tech. Rep. CMU/SEI-2003-TN-033, Software Engineering Institute, 2003.

[49] G. Macher, E. Armengaud, E. Brenner, and C. Kreiner, "Threat and Risk Assessment Methodologies in the Automotive Domain," in *Proceedings of the 7th International Conference on Ambient Systems, Networks and Technologies (ANT 2016) / The 6th International Conference on Sustainable Energy Information Technology (SEIT-2016) / Affiliated Workshops*, vol. 83 of *Procedia Computer Science*, pp. 1288–1294, Elsevier, 2016.

[50] ISO, *ISO 26262: Road Vehicles – Functional Safety*, 2011.

[51] IEC, *IEC 61025 Fault Tree Analysis (FTA)*, 2006.

[52] IEC, *IEC 60812 Analysis Techniques for System Reliability-Procedure for Failure Mode and Effects Analysis (FMEA)*, 2006.

[53] IEC, *IEC 61882 Hazard And Operability Studies (HAZOP Studies)-Application Guide*, 2001.

[54] C. A. Ericsson, *Hazard Analysis Techniques for System Safety*, John Wiley & Sons, 2015.

[55] N. G. Leveson, *Engineering A Safer World: Systems Thinking Applied to Safety*, MIT Press, 2011.

[56] "Introduction to Microsoft Security Development Lifecycle (SDL) Threat Modeling," https://www.microsoft.com/en-us/sdl/default.aspx.

[57] B. Schneier, "Attack Trees: modeling security threats," *Dr. Dobb's Journal*, vol. 24, no. 12, pp. 21–29, 1999.

[58] B. Kordy, S. Mauw, S. Radomirovic, and P. Schweitzer, "Foundation of Attack-Defense Trees," in *Proceedings of the International Workshop on Formal Aspects in Security and Trust*, Springer, 2010.

[59] A. Roy, D. S. Kim, and K. S. Trivedi, "Attack countermeasure trees (ACT): Towards unifying the constructs of attack and defense trees," *Security and Communication Networks*, vol. 5, no. 8, pp. 929–943, 2012.

[60] G. Sindre and A. L. Opdahl, "Eliciting security requirements with misuse cases," *Requirements Engineering*, vol. 10, no. 1, pp. 34–44, 2005.

[61] J. P. McDermott and C. Fox, "Using abuse case models for security requirements analysis," in *Proceedings of the 15th Annual Computer Security Applications Conference, IEEE Computer Society*, pp. 55–64, Phoenix, Ariz, USA, 1999.

[62] T. Okubo, K. Taguchi, H. Kaiya, and N. Yoshioka, "MASG: Advanced misuse case analysis model with assets and security goals," *Journal of Information Processing*, vol. 22, no. 3, pp. 536–546, 2014.

[63] UK Department for Transport, *Rail Cyber Security Guidance to Industry*, 2016.

[64] K. Netkachova and R. E. Bloomfield, "Security-Informed Safety," *The Computer Journal*, vol. 49, no. 6, pp. 98–102, 2016.

[65] RTCA, *DO-326A Airworthiness Security Process Specification*, 2014.

[66] SESAMO, "Security and Safety Modelling," 2015, http://sesamo-project.eu.

[67] S. Kriaa, L. Pietre-Cambacedes, M. Bouissou, and Y. Halgand, "A survey of approaches combining safety and security for industrial control systems," *Reliability Engineering & System Safety*, vol. 139, pp. 156–178, 2015.

[68] S. Kriaa, *Joint Safety and Security Modeling for Risk Assessment in Cyber Physical Systems [Diss.]*, Université Paris-Saclay, 2016.

[69] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[70] N. G. Leveson, *Safeware: System Safety and Computers*, Addison-Wesley, 1995.

[71] C. Schmittner and Z. Ma, "Towards a Framework for Alignment Between Automotive Safety and Security Standards," in *Proceedings of the SAFECOMP*, LNCS 9338, pp. 133–143, 2015.

[72] D. Firesmith, "Specifying reusable security requirements," *The Journal of Object Technology*, vol. 3, no. 1, pp. 61–75, 2004.

[73] ISO/IEC, *ISO/IEC 27005: 2011 Information Technology – Security Techniques – Information Security Risk Management*, 2011.

[74] ISO, *ISO 31000: Risk Management – Principles And Guidelines*, 2009.

[75] ITU, *ITU-T X.1524: Cybersecurity Information Exchange –Vulnerability/State Exchange, Common Weakness Enumeration*, 2012.

[76] ITU, *ITU-T X.1521: Cybersecurity Information Exchange, Vulnerability/State Exchange, Common Vulnerability Scoring System*, 2011.

[77] Y. Miyashita, H. Yasukawa, I. Kurosaki, T. Matsuo, S. Horihata, and N. Kobayashi, "On-vehicle compact and lightweight multi-channel central gateway unit," *SEI Technical Review*, no. 83, pp. 5–9, 2016.

[78] D. Davidson, W. Hao, R. Jellinek, T. Ristenpart, and V. Singh, "Thomas Ristenpart, and Vikas Singh, Controlling uavs with

sensor input spoofing attacks," in *Proceedings of the 10th USENIX Workshop on Offensive Technologies, (WOOT 16)*, N. Silvanovich and P. Traynor, Eds., pp. 221–231, 2016.

[79] M. Lichtman, J. D. Poston, S. Amuru et al., "A Communications Jamming Taxonomy," *IEEE Security & Privacy*, vol. 14, no. 1, pp. 47–54, 2016.

[80] D. He, S. Chan, and M. Guizani, "Communication Security of Unmanned Aerial Vehicles," *IEEE Wireless Communications Magazine*, vol. 24, no. 4, pp. 134–139, 2017.

Hindawi
Security and Communication Networks
Volume 2019, Article ID 1719585, 13 pages
https://doi.org/10.1155/2019/1719585

Research Article

Single-Round Pattern Matching Key Generation Using Physically Unclonable Function

Yuichi Komano,[1] Kazuo Ohta,[2] Kazuo Sakiyama,[2] Mitsugu Iwamoto,[2] and Ingrid Verbauwhede[3]

[1] *Toshiba Corporation, Kawasaki, Japan*
[2] *The University of Electro-Communications, Tokyo, Japan*
[3] *KU Leuven, Leuven, Belgium*

Correspondence should be addressed to Yuichi Komano; yuichi1.komano@toshiba.co.jp

Received 6 July 2018; Revised 9 November 2018; Accepted 12 December 2018; Published 1 January 2019

Guest Editor: Daniel Schneider

Copyright © 2019 Yuichi Komano et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Paral and Devadas introduced a simple key generation scheme with a physically unclonable function (PUF) that requires no error correction, e.g., by using a fuzzy extractor. Their scheme, called a pattern matching key generation (PMKG) scheme, is based on pattern matching between auxiliary data, assigned at the enrollment in advance, and a substring of PUF output, to reconstruct a key. The PMKG scheme repeats a round operation, including the pattern matching, to derive a key with high entropy. Later, to enhance the efficiency and security, a circular PMKG (C-PMKG) scheme was proposed. However, multiple round operations in these schemes make them impractical. In this paper, we propose a single-round circular PMKG (SC-PMKG) scheme. Unlike the previous schemes, our scheme invokes the PUF only once. Hence, there is no fear of information leakage by invoking the PUF with the (partially) same input multiple times in different rounds, and, therefore, the security consideration can be simplified. Moreover, we introduce another hash function to generate a check string which ensures the correctness of the key reconstruction. The string enables us not only to defeat manipulation attacks but also to prove the security theoretically. In addition to its simple construction, the SC-PMKG scheme can use a weak PUF like the SRAM-PUF as a building block if our system is properly implemented so that the PUF is directly inaccessible from the outside, and, therefore, it is suitable for tiny devices in the IoT systems. We discuss its security and show its feasibility by simulations and experiments.

1. Introduction

The Internet of Things (IoT) is widely spread to make us more intelligent, efficient, and comfortable. In IoT systems, devices are located everywhere to exchange their sensing data and their control information. On the other hand, lots of devices in these systems are resource-constrained where it is hard to implement security functions. Unlike the closed system with a limited number of devices, in the IoT systems, attackers are able to obtain devices to analyze them maliciously, and therefore they can be weak points of these systems.

Let us consider the safety of the IoT systems. First of all, the reliability of data is important because the devices work unwillingly with the improper data, especially with the data manipulated by the attacker. Moreover, the correctness of the firmware including the safety functionalities, such as the fail-stop and the fail-tolerance, is also important. In order to avoid the manipulation of the firmware by attacker so that the safe functionality does not work, the firmware should be well protected and securely updated. Both for the data reliability and for the firmware correctness, the security is essential. Hence, securing such devices is one of our emerging challenges. Particularly, the management of key is a crucial task.

The physically unclonable function (PUF) is one of the promising primitives to improve the security of tiny devices. It derives a unique value for each device (function) from its fine characteristics. For example, the SRAM-PUF [1–3] uses initial states of the SRAM cells just after the power-on as such characteristics; and, the Arbiter-PUF [4–6] uses the logic delays of a dual-rail circuit as such ones. The unique value derived is used as (a source of) an identity (ID) or a

cryptographic key. The PUF is suitable for a tiny device for the following two reasons. First, since these characteristics naturally arise during the manufacturing process, we can remove the process of embedding an ID or a key and decrease the manufacturing cost. Second, since analyzing the unique and fine characteristics is difficult, it can be regarded as a secure and tamper- evident storage which proves the physical manipulation for analyzing the key.

On the other hand, the output of PUF may include a small noise for each invocation, e.g., 5% or up to 15% noise in the signal, because of external factors such as external/ambient temperature variations and/or supplied voltage variations. In order to use the output as a cryptographic key, the noise should be removed by error correction techniques, such as the fuzzy extractor [7, 8], because the cryptographic results completely differ if the key differs by only one bit. The fuzzy extractor, however, includes a complex error correction code whose cost might be high for the tiny devices.

1.1. Pattern Matching Key Generation Using PUF. Paral and Devadas [9] gave an interesting solution to remove the noises without an error correcting code. Their proposal uses a pattern matching which only estimates the Hamming distance between two strings. Their main idea is to regard the index (bit position) indicating a substring of the PUF output as a secret, instead of the (sub)string of PUF output itself. This scheme is called a pattern matching key generation (PMKG) scheme.

The PMKG scheme consists of multiple rounds. In each round of its enrollment phase, a key generation device extracts a substring of PUF output indicated by a secret index and stores the substring into the public (and maybe insecure) nonvolatile memory (NVM) area, as auxiliary data. The PMKG scheme regards a (hash value of a) concatenation of the substrings as a secret key. In each round of its reconstruction phase, the device compares a noisy PUF output with the stored auxiliary data and recovers the corresponding index and eventually the key.

Against the PMKG scheme, Delvaux and Verbauwhede [10, 11] gave attacks, named *snake attacks*, to recover the secret indices. Their attacks modify the auxiliary data stored in the NVM bit by bit in each round and detect the index by running the device with the modified data. To avoid the snake attacks, they also suggested regarding the PUF output as circular data. We call the scheme a circular PMKG (C-PMKG) scheme.

The PMKG (C-PMKG, respectively) scheme stores, for each round, the substring (resp., circularly shifted string) of PUF output into the NVM as auxiliary data. If two auxiliary data pieces for different rounds have the same substring (intersection), it might leak information on the secret indexes for the corresponding rounds. To mitigate the leakage, there are several solutions. The first one is to control the inputs of PUF for rounds to avoid such intersection. Another one is to enlarge the input/output space so as to neglect the intersection. These solutions increase the implementation cost and may be inappropriate for tiny devices.

1.2. Our Contribution. In this paper, we propose a simple and efficient PMKG scheme, named a single-round circular PMKG (SC-PMKG) scheme. Unlike the previous schemes, the SC-PMKG scheme consists of a single-round operation with multiple indexes. It divides a PUF output into multiple substrings, circularly shifts each substring by each secret index, and stores the shifted data into the NVM as auxiliary data. Moreover, it also stores a hash value, as a check string to check the correctness of the key reconstruction, into the NVM.

By reducing the number of rounds in the (C-)PMKG scheme to one, we need not care for the information leakage from the PUF output strings in the auxiliary data with the (partially) same input, and therefore the security discussion can be simplified. In addition, the check string not only disables the manipulation attacks but also ensures the provable security of our scheme in the random oracle model [12]. We give a security proof under the condition (assumption), for simplicity, that there is no adversarial interface to access the internal PUF. This condition also provides our choice of the PUF candidates applicable to the SC-PMKG scheme. Without such interfaces, machine learning attacks to the PUF are impractical, and, hence, various PUFs can be applicable to the SC-PMKG, including a weak PUF such as the SRAM-PUF. (The modeling attack requires challenge-response pairs (CRPs) of PUF, which is applicable to PUF applications where adversaries can obtain the PUF CRPs, such as an authentication with PUF. In the application of the key generation, adversaries can obtain the PUF outputs from auxiliary data. Against this application, the modeling attack could be meaningless for the following two reasons. First, the corresponding inputs for the auxiliary data are unknown to adversaries. Second, the number of auxiliary data is limited; for example, if the key generation device is specific to a key of certain application, adversaries can obtain PUF outputs only for the key.) Note that, on the provable security, we can relax the assumption where there are adversarial interfaces to the PUF *except* one to directly access the output corresponding to the target key, as we discuss in Section 4.3. Such discussions on the adversarial conditions and the security show that the security coengineering for the developments of the PUF devices and of the PUF application is important.

Furthermore, we check its feasibility by simulations and experiments. We also check the validity with theoretical estimation. Our theoretical estimation can derive parameter candidates not only for the SC-PMKG scheme but also for the C-PMKG schemes. From our discussion, the SC-PMKG scheme is secure and efficient rather than the previous PMKG schemes; and, therefore, it is suitable for tiny devices to tackle the emerging threats against the IoT systems.

1.3. Organization. In Section 2, we give the definitions of PUF and PUF-based key generation scheme. We then review the constructions of previous PMKG schemes and discuss their drawbacks in Section 3. In Section 4, we present the SC-PMKG scheme and discuss its security. We then test its feasibility and make comparisons among the PMKG schemes in Sections 5 and 6, respectively. Finally, Section 7 concludes this paper.

2. Definition

In this section, we review definitions of the PUF and PUF-based key generation scheme.

2.1. PUF. References, such as [13, 14], gave a definition of the PUF with several properties. In this paper, we regard the PUF as a physical function satisfying relaxed properties below, in order for various types of PUF to be applicable to our PMKG.

Definition 1 (PUF). A family of physical functions is called the family of PUFs if these functions satisfy the following properties.

(1) Each function works within polynomial steps

(2) Each function produces an output including a small noise; however, the signal-to-noise ratio is high enough to remove the noise by using an error correcting code. Hence, it acts as a function which provides a particular output for an input

(3) It is practically hard to characterize or clone each function

(4) Functions produce unique outputs per each of them

The third property does not ensure that it is hard to characterize or clone the physical function. The function may be characterized or cloned in future, if more sophisticated physical equipment to analyze it appears. In this paper, we regard the SRAM-PUF, Arbiter-PUF, and ring-oscillator PUF (RO-PUF) with fine processes and careful designs as PUFs.

We give a security assumption of PUF, which is a formal description of above third condition, to be used in the security proof of the SC-PMKG scheme.

Definition 2 (indistinguishable PUF). Let ℓ_c and ℓ_p be input and output lengths of PUF, respectively. We state that a physically unclonable function PUF is indistinguishable in (τ, q, ϵ) if any adversary $\mathscr{A}$, within the time bound τ, cannot achieve an advantage more than ϵ, even if $\mathscr{A}$ observes q input-output pairs $\{(x_i, y_i)\}$. The advantage is defined with $\mathsf{Adv}_{\mathscr{A}}^{\mathsf{Ind-PUF}} = \Pr[\mathscr{A}(y) = 1 \mid \mathscr{A}(1^{\ell_c}, 1^{\ell_p}) = x; y = \mathsf{PUF}(x)] - \Pr[\mathscr{A}(y) = 1 \mid \mathscr{A}(1^{\ell_c}, 1^{\ell_p}) = x; y \longleftarrow \{0,1\}^{\ell_p}]$. Here, $\mathscr{A}$ is allowed to output x where there is no overlap between $\mathsf{PUF}(x)$ and the observed PUF outputs $\mathsf{PUF}(x_i)$ for $i \in [1, q]$.

This indistinguishability captures the unpredictability and unclonability, which relate to the third property of Definition 1. Intuitively, the advantage $\mathsf{Adv}_{\mathscr{A}}^{\mathsf{Ind-PUF}}$ is a metric of adversary's ability to distinguish the PUF output from a random string. The distinguishing game is as follows: The adversary $\mathscr{A}$ chooses an input x for the PUF and receives either an output $y = \mathsf{PUF}(x)$ or a random string $y \in \{0,1\}^{\ell_p}$. $\mathscr{A}$ outputs 1 if $\mathscr{A}$ supposes that $y = \mathsf{PUF}(x)$. In this game, $\mathscr{A}$ may collect input-output pairs of PUF, as hints, by accessing the PUF oracle.

Let us assume the SRAM-PUF. In the SRAM-PUF, the challenge is an ℓ_c-bit address x, and the output y is a sequence of initial states of ℓ_p SRAM cells starting with an address x. Definition 2 allows $\mathscr{A}$ to collect a pair $\{(x_i, y_i)\}$ for $1 \le i \le q$. With these pairs, the $\mathscr{A}$'s restriction stated in Definition 2 means that $\mathscr{A}$ is disallowed to output x where any subsequence of $y = \mathsf{PUF}(x)$ is included in $\{y_i\}_{i \in [1,q]}$.

2.2. PUF-Based Key Generation. We then review a definition of the PUF-based key generation (PBKG) scheme.

Definition 3 (PUF-based key generation, PBKG). A PBKG scheme consists of the following two phases.

(i) **Enrollment phase**: A key generation device in the PBKG scheme assigns a key and auxiliary data which help to reconstruct the key from a noisy PUF output in the next phase. The auxiliary data are stored into a public (maybe insecure) nonvolatile memory (NVM). Note that the device can generate the key inside itself or receive the key from outside.

(ii) **Reconstruction phase**: The key generation device regenerates the PUF output with some noises, loads the auxiliary data from the NVM, and reconstructs the key from these data.

We then review a definition of security for the PBKG scheme. As Datta et al. [15] discussed, the indistinguishability [16, 17] may be inadequate if the PBKG scheme is composed of another cryptographic protocol, such as an encryption scheme or message authentication code (MAC). This is because even though a key derived from the PBKG scheme is indistinguishable from a random string, the indistinguishability can be lost when the key is used, e.g., to encrypt a (partially) known message. In order to ensure the security for a combination of the PBKG scheme and a cryptographic protocol, we should customize the definition of its security by modeling an adversary to collect hints from both the PBKG scheme and the cryptographic protocol; namely, to access oracles of the scheme and the protocol, respectively. In this paper, we propose a PBKG scheme for a general purpose. Therefore, we discuss the indistinguishability [16, 17] as follows.

Definition 4 (indistinguishable PBKG). Let ℓ_k be a bit length of a key derived from a PBKG scheme. We say that the PBKG scheme is indistinguishable in (τ, q, ϵ) if any adversary $\mathscr{A}$, within the time bound τ, cannot achieve an advantage more than ϵ even if $\mathscr{A}$ invokes a key generation device with a (maybe intentionally modified) auxiliary data at most q times. Here, the advantage is defined with $\mathsf{Adv}_{\mathscr{A}}^{\mathsf{pbkg}} = |p_1 - p_2|$, where $p_1 = \Pr[\mathscr{A}(y) = 1 \mid y \longleftarrow \mathsf{PBKG}(1^{\ell_k})]$ and $p_2 = \Pr[\mathscr{A}(y) = 1 \mid y \longleftarrow \{0,1\}^{\ell_k}]$.

3. Previous Works and Their Drawbacks

This section reviews the PMKG and C-PMKG schemes and discusses their drawbacks.

3.1. PMKG Scheme. Figures 1 and 2 show the building blocks of the PMKG scheme [9] in its enrollment and reconstruction

phases, respectively. They use the following functions.

(i) PUF: Physically unclonable function which, given an ℓ_c-bit input x, outputs an ℓ_p-bit string y

(ii) CS: Challenge sequencer which, given at least one $\ell_i(= \lceil \log_2 \ell_p \rceil)$-bit index ind and a flag $flag$ as inputs, outputs the input of PUF x

(iii) KGF: Key generation function which, given a set of ℓ_i-bit indexes $\{ind_i\}_i$, outputs an ℓ_k-bit key key

(iv) pattern match: Comparator which, given ℓ_w-bit ($\ell_w < \ell_p$) auxiliary data stored in a public NVM and an ℓ_p-bit PUF output, looks for an index that leads an ℓ_w-bit substring of PUF output near to the auxiliary data. It may look for an index with which the distance between auxiliary data and substring indicated by the index is less than some predetermined threshold and/or with which the distance is the smallest among all substring candidates

Paral and Devadas [9] used the Arbiter-PUF as a building block which outputs only one bit per an input. In order to obtain an ℓ_p-bit output with the Arbiter-PUF, CS generates ℓ_p inputs $\{x_j\}_{j\in[1,\ell_p]}$ sequentially from index(es), instead of one input x as above, within each round. And then PUF returns ℓ_p-bit output in total with the ℓ_p inputs in the round. More precisely, we extend the above notations; we extend CS to output l_p inputs (by regarding x as $\{x_j\}_{j\in[1,\ell_p]}$), and we extend PUF to output an ℓ_p-bit concatenation $\mathsf{PUF}(x_1)\|\cdots\|\mathsf{PUF}(x_{\ell_p})$ for inputs $\{x_j\}_j$. Note that the above notations naturally cover the SRAM-PUF. Throughout this paper, for simply, we use the above notations regardless of the type of PUF.

The PMKG scheme uses four registers: an input register, an output register, an index register, and a key register, as depicted in Figures 1 and 2. The input register holds an input of PUF. The output register holds an output of PUF and it is randomly accessible. The index register holds a set of indexes. The key register holds a recovered key to pass it to the cryptographic protocol.

The PMKG scheme also uses the public NVM. In the enrollment phase, a key generation device stores a set of substrings of PUF outputs as a set of auxiliary data into the NVM. In the reconstruction phase, the key generation device loads the auxiliary data and uses it to reconstruct the key. Note that the NVM may be insecure and the stored auxiliary data may be modified by an adversary.

Protocol 1 (PMKG). Let PUF, CS, KGF, and pattern match be functions as above. A PMKG scheme consists of two phases: an enrollment phase and a reconstruction phase below.

Enrollment phase: A key generation device repeats a round operation r times with indexes $\{ind_i\}_{i\in[0,r]}$. The index ind_0 is an initial input of CS and we assume that it is fixed and stored into the NVM in advance. (Since the NVM is public, we assume that ind_0 is observable by adversaries. At the final steps of enrollment and reconstruction phases, we remove ind_0 from the input of KGF, because its observable value does not increase the entropy of a key.) Other indexes may be also fixed in advance or randomly generated inside the device, and we assume the former case here. This phase consists of the following steps.

(1) The key generation device sets $flag$ to one

(2) For $i \in [1, r]$, the device repeats the following steps:

(a) It inputs $(\{ind_k\}_{k<i}, flag)$ into CS to generate an input of PUF x_i

(b) It inputs x_i into PUF to generate a PUF output y_i

(c) It stores a substring of PUF output $z_i = y_i[ind_i : ind_i + \ell_w - 1]$ into the NVM. Here, $s[a : b]$ denotes a substring of s, consisting of its a-th to b-th bits

(3) It regards $\mathsf{KGF}(\{ind_i\}_{i\in[1,r]})$ as key

Reconstruction phase: The key generation device repeats a round operation r times to recover the indexes and eventually the key, assigned at the enrollment phase. Note that a PUF output in this phase may differ from that in the enrollment phase. We add prime marks to the variables in this phase. This phase consists of the following steps.

(1) The key generation device sets $flag$ to one

(2) For $i \in [1, r]$, the device repeats the following steps:

(a) It inputs $(\{ind'_k\}_{k<i}, flag)$ into CS to generate an input of PUF x'_i

(b) It inputs x'_i into PUF to generate a PUF output y'_i

(c) It loads auxiliary data z_i from the NVM

(d) It looks for an index ind'_i indicating the substring of y'_i (from ind'_i-th bit to $(ind'_i + \ell_w - 1)$-th bit) near to z_i from the pattern match. If a mismatch happens, namely, if such an index is not detected or if more than one index is detected, ind'_i and $flag$ are set with a constant value, e.g., ℓ_w and zero, respectively

(3) If $flag = 1$, then it inputs $\{ind'_i\}_{i\in[1,r]}$ into KGF to recover key. Otherwise, it aborts.

Paral and Devadas [9] regarded PUF as a 4-XOR PUF which blends outputs of four independent Arbiter-PUFs. The blended PUF decreases a correlation between inputs and outputs of PUF; and therefore it makes the machine learning attacks [18] meaningless. The more the number of blend increases, the more difficult the attack is. In this paper, we assume that the PUF is ideal and indistinguishable as in Definition 2 where there is no correlation between the inputs and the outputs. To realize such PUF, the blending is one of solutions.

The PMKG scheme has several drawbacks as follows. First of all, it requires lots of PUF output, and therefore it may be unsuitable for tiny devices in the IoT systems. The key generation device regards the ℓ_w-bit substring out of the ℓ_p-bit PUF output as auxiliary data. In other words, it discards

the remaining $(\ell_p - \ell_w)$-bit in each round. In [9], they assumed that, as an example, $\ell_p = 1279, \ell_w = 256$ and $r = 16$. With this example, the $(1279 - 256) \times 16 = 16368$ bits, out of $1279 \times 16 = 20464$ PUF output bits, are discarded. It makes the PMKG scheme inefficient, by requiring the execution time to generate the 16368 PUF output bits and the power consumption more.

In addition to its inefficiency, it has a security vulnerability, caused by attacks with an NVM manipulation. Such a manipulation leads mismatches in its reconstruction and these mismatches change the device's behavior. Delvaux and Verbauwhede [10, 11] gave attacks, named *snake attacks*, which modify the auxiliary data in the NVM, invoke the key generation with the modified one, and guess the index ind_i by observing its behavior.

Specifically, their attacks repeatedly modify the auxiliary data in the NVM to be (noncircularly) one bit shifted by guessing the next bit of PUF output. If the modified auxiliary data move out of the range of the PUF output with a large amount of shift, a mismatch happens and the device's behavior changes; until the amount is less than or equal to the distance to head or end, the device outputs a key, which should be different from the assigned one because inputs for CS and KGF differ; on the other hand, if it exceeds the distance, the device aborts because the mismatch happens. Namely, the snake attacks repeat the shift and guess by increasing its amount of shift until the device's behavior changes and return the amount as its guess for the secret index.

3.2. C-PMKG Scheme. To enhance the efficiency and/or the security, a circular PMKG (C-PMKG) scheme was proposed in [10, 19] independently and further discussed in [11, 20], respectively. It regards PUF outputs circularly shifted by secret indices $\{ind_i\}$ as auxiliary data.

Figures 3 and 4 show its building blocks. As for PUF, CS, pattern match, and KGF, we use the same notations as ones for the PMKG scheme, while we set $\ell_w = \ell_p$ in PUF. In addition to them, it also uses the function rotate:

(i) rotate: rotate function which, given an ℓ_w-bit string y and an amount of circularity $ind \in [0, \ell_w - 1]$, circularly shifts y by ind-bit

Protocol 2 (C-PMKG). Let PUF, CS, pattern match, and KGF be functions as in the PMKG scheme, except that the output length of the PUF is not ℓ_p-bit but ℓ_w-bit. Also let rotate be a function as above. The C-PMKG scheme consists of following two phases: an enrollment phase and a reconstruction phase below.

Enrollment phase: A key generation device repeats a round operation r times with indexes $\{ind_i\}_{i\in[0,r]}$. The index ind_0 is an initial input of CS and we assume that it is fixed in advance and stored in the NVM. Other indexes may be also fixed in advance or randomly chosen inside the device, and we assume the former case. This phase consists of the following steps.

(1) The key generation device sets *flag* to one

(2) For $i \in [1, r]$, the device repeats the following steps:

(a) It inputs $(\{ind_k\}_{k<i}, flag)$ into CS to generate an input of PUF x_i

(b) It inputs x_i into PUF to generate a PUF output y_i

(c) It stores a circularly shifted PUF output $z_i = \mathsf{rotate}(y_i, ind_i)$ into the NVM as auxiliary data

(3) It regards $\mathsf{KGF}(\{ind_i\}_{i\in[1,r]})$ as *key*

Reconstruction phase: The key generation device repeats a round operation r times to recover the indexes and eventually key, assigned at the enrollment phase. This phase consists of the following steps.

(1) The key generation device sets *flag* to one

(2) For $i \in [1, r]$, the device repeats the following steps:

(a) It inputs $(\{ind'_k\}_{k<i}, flag)$ into CS to generate an input of PUF x'_i

(b) It inputs x'_i into PUF to generate a PUF output y'_i

(c) It loads auxiliary data z_i from the NVM

(d) It looks for an index ind'_i indicating the circularity shifted string $\mathsf{rotate}(y'_i, ind'_i)$ near to z_i from the pattern match. If a mismatch happens, namely, if such an index is not detected or if more than one index is detected, ind'_i and $flag$ are set with a constant value ℓ_w and zero, respectively

(3) If $flag = 1$, then it inputs $\{ind'_i\}_{i\in[1,r]}$ into KGF to recover *key*. Otherwise, it aborts

Unlike the PMKG scheme, the C-PMKG scheme uses the whole (circularly shifted) PUF output as auxiliary data. Removing the discarding of PUF outputs enhances its efficiency compared to the PMKG scheme. Moreover, since there is no head or end in the circularity, the snake attacks are meaningless to the C-PMKG scheme.

However, in order for the C-PMKG scheme to be in use, there are technical issues to be considered. The most important one is a careful design of CS. The circularly shifted PUF outputs are stored in the public NVM as auxiliary data in the C-PMKG scheme; and, here, if some of them have intersections, the information of the corresponding indexes may be leaked. Hence, to ensure the security of C-PMKG scheme, CS should be designed leading to no intersection over rounds.

4. Single-Round Circular PMKG Scheme

In this section, we propose a simple PMKG scheme, named single-round circular PMKG (SC-PMKG), and show its security against manipulation attacks.

4.1. Our Idea. We construct the SC-PMKG scheme, based on the C-PMKG scheme, by improving the simplicity and security as follows.

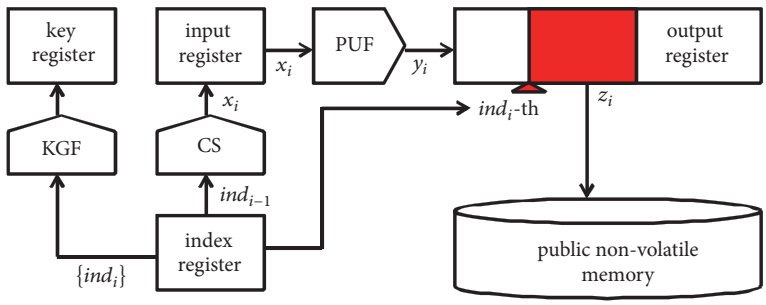


Figure 1: Building blocks for enrollment phase in PMKG.

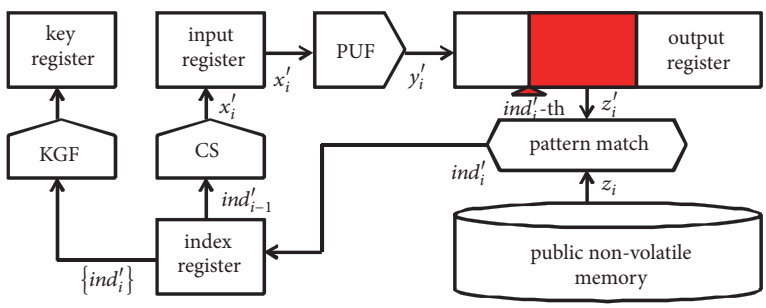


Figure 2: Building blocks for reconstruction phase in PMKG.

(i) *Simplicity*: Unlike the previous PMKG schemes with r round operations, the SC-PMKG scheme is performed within a single-round. In order to increase the entropy of key efficiently, the previous schemes prepare r PUF output strings with ℓ_w-bit each. On the other hand, the SC-PMKG scheme prepares an $(n \cdot \ell_w)$-bit PUF output string and divides it into n substrings with ℓ_w-bit each. From this change, there is no fear to overlap among the (sub)strings and the security discussion can be simplified. Moreover, the simple design helps us to develop devices.

(ii) *Security*: To detect and defeat the manipulation attack, we introduce a check string cs to confirm the correctness of the key reconstruction. The string is derived from a cryptographic hash function with which, by regarding it as a random oracle, we can theoretically prove the security of the SC-PMKG scheme.

The following subsections give our construction and the security consideration.

4.2. Construction. As for PUF, CS, rotate, pattern match, and KGF, we use the same notations for the C-PMKG scheme, except that we assume $\ell_p > \ell_w$ for PUF as in the PMKG scheme. Additionally, our scheme uses a hash function hash, to check the integrity of a reconstructed key.

(i) hash: Cryptographic one-way hash function which, given an ℓ_k-bit key, returns an ℓ_{cs}-bit check string cs

Protocol 3 (SC-PMKG). Let PUF, CS, rotate, pattern match, and KGF be functions as in the C-PMKG scheme, except that we assume $\ell_p = n \cdot \ell_w$ for an integer n, and let hash be a hash function as above. The SC-PMKG scheme consists of the following two phases: an enrollment phase and a reconstruction phase below.

Enrollment phase: A key generation device with indexes $\{ind_i\}_{i\in[0,n]}$ performs an enrollment as follows. The index ind_0 is an initial input for CS and we assume that it is fixed in advance and stored in the NVM. Other indexes may be also fixed in advance or randomly chosen inside the device, and we assume the former case. This phase consists of the following steps.

(1) The key generation device inputs ind_0 into CS to generate an input of PUF x

(2) It inputs x into PUF to generate a PUF output

(3) It divides the PUF output into n substrings $\{y_i\}_{i\in[1,n]}$ with ℓ_w-bit each

(4) It stores $\{z_i = \mathsf{rotate}(y_i, ind_i)\}$ for $i \in [1, n]$ into the NVM as auxiliary data

(5) It computes $key = \mathsf{KGF}(\{ind_i\}_{i\in[1,n]})$

(6) It computes $cs = \mathsf{hash}(ind_0, \{(z_i, ind_i)\}_{i\in[1,n]}, key)$ and stores it into the NVM as a check string

Reconstruction phase: The key generation device recovers the indexes and eventually the key, assigned at the enrollment phase. This phase consists of the following steps.

(1) The device sets $flag$ to one

(2) It inputs ind'_0 into CS to generate an input of PUF x'

(3) It inputs x' into PUF to generate a PUF output

(4) It divides the PUF output into n substrings $\{y'_i\}_{i\in[1,n]}$ with ℓ_w-bit each

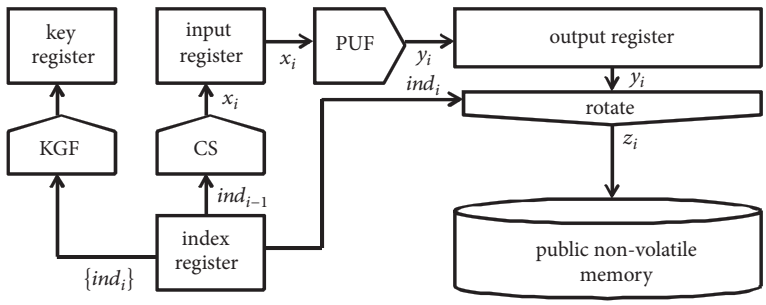


Figure 3: Building blocks for enrollment phase in C-PMKG.

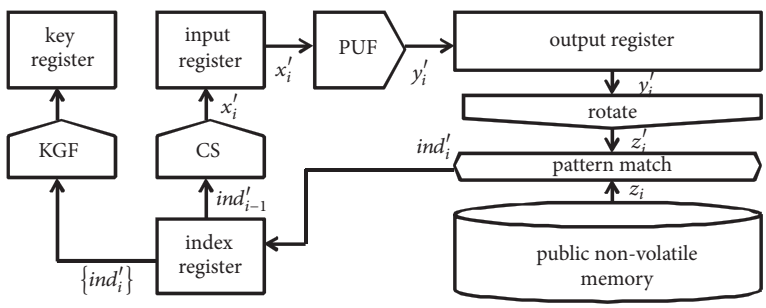


Figure 4: Building blocks for reconstruction phase in C-PMKG.

(5) For $i \in [1, n]$, it repeats the following steps:

(a) It loads auxiliary data z_i' from the NVM

(b) It looks for an index ind_i' indicating the circularity shifted string $\mathsf{rotate}(y_i', ind_i')$ near to z_i from the $\mathsf{pattern\ match}$. If a mismatch happens, ind_i' and $flag$ are set with a constant value ℓ_w and zero, respectively

(6) If $flag = 1$, then it inputs $\{ind_i'\}_{i\in[1,n]}$ into KGF to compute key'. Otherwise, it aborts

(7) It loads a check string cs' from the NVM

(8) If $\mathsf{hash}(ind_0', \{(z_i', ind_i')\}_{i\in[1,n]}, key') = cs'$ holds, it regards key' as the reconstructed key. Otherwise, it aborts

There are two advantages in the SC-PMKG scheme. The first one is that, from its simple construction with a single-round, we do not need to pay attention to the intersections of auxiliary data over rounds. It simplifies the design of CS and the security consideration. The second one is that, by introducing the check string, the security against manipulation attacks of the NVM is ensured, if hash is an ideal hash function (random oracle [12]). We discuss the security in the following subsection.

4.3. Security Consideration. As in the C-PMKG scheme, since the auxiliary data is obtained from the circularly shift of the PUF output, the snake attacks are invalid for the SC-PMKG scheme. In addition, we can prove that the SC-PMKG scheme is an indistinguishable PBKG scheme in the random oracle model [12]. As for the security, the following theorem holds.

Theorem 5. *Assume that* PUF *is an indistinguishable PUF in* $(\tau, 0, \epsilon)$ *and that* hash *is the random oracle. Then, the SC-PMKG scheme is an indistinguishable PBKG in* (τ', q_K, ϵ') *where*

$$\epsilon' = \epsilon - \frac{1}{\ell_w^n}$$
$$\text{and } \tau' = \tau + q_K O(\mathsf{T}_K). \tag{1}$$

Here, T_K *denotes an execution time of the SC-PMKG scheme.*

The proof is done by the contradiction. Namely, we show that if there exists an adversary $\mathscr{A}$ against the SC-PMKG scheme, we can construct an algorithm $\mathscr{B}$, which uses $\mathscr{A}$ as a subroutine, to break the indistinguishability of PUF. The construction of $\mathscr{B}$ is simple and we give a proof in the Appendix.

Note that, in Theorem 5, we assume that $(\tau, 0, \epsilon)$ for the underlying PUF. The condition of $q = 0$ makes the PUF strongly protected within the system, lest no PUF output for an adversarial input is observable. However, there is a trade-off between the assumptions for the secure system ($q = 0$) and for a secure PUF ($q > 0$, see next paragraph). The condition we assume relaxes the choice of PUF because we can neglect the fear of the machine learning attacks [18], and because we can use a weak PUF with a small input-output space, such as SRAM-PUF.

We can extend the security model of Definition 4 so that an adversary against the PBKG scheme is allowed to

collect input-output pairs of the underlying PUF; that is, we can consider the security model with a powerful adversary. In this case, the indistinguishability of the PBKG scheme is defined with four-tuple $(\tau', q_K, q_P, \epsilon')$, instead of three-tuple (τ', q_K, ϵ'), where the adversary accesses the PBKG and PUF oracles at most q_K and q_P times, respectively. With this extension, we can prove another theorem, if the underlying PUF is indistinguishable in (τ, q_P, ϵ), instead of $(\tau, 0, \epsilon)$. If q_K is so large, a weak PUF may not be a possible choice for our system. Note that, even with this extension, the adversary of the SC-PMKG scheme should be restricted not to access the PUF oracle on x corresponding to the target key. To simplify the security proof, this paper discusses the security without the extension.

Regardless of whether the model is extended or not, we require the PUF so that its output is unobservable by invalid interface other than the interfaces in Definition 2. As for the SRAM-PUF, for example, to avoid the physical probing attack [21, 22], we should choose an appropriate PUF, with a fine process [23], etc.

4.4. Notes on Parameters. The SC-PMKG scheme is parameterized with the following parameters: ℓ_w, n, $\ell_p = n \cdot \ell_w$, $\ell_i = \log_2 \ell_w$, and ℓ_{cs}. If the pattern match in the reconstruction phase looks for the index with a threshold, the threshold th is also required.

Among them, n and ℓ_i (and, therefore, ℓ_w and ℓ_p) relate to the security. The (Shannon) entropy of key source is estimated by $n \cdot \ell_i$. To ensure the 160-bit entropy for the source, we set them so that $n \cdot \ell_i$ exceeds 160. In addition, ℓ_{cs} also relates to the security. The larger it is, up to $n \cdot \ell_i$, the harder the manipulation attack is.

On the other hand, ℓ_w and th relate to the correctness. There are two failure scenarios in reconstructing the key: the pattern match fails at Step (5)(b) or the hash value does not match the check string at Step (8). Note that if the indexes are correctly recovered at Step (5)(b), the verification at Step (8) should succeed. Hence, let us discuss the failure at Step (5)(b).

The pattern match at Step (5)(b) looks for an index with which the distance between a substring candidate and the auxiliary data is less than th and/or the smallest, as stated in Section 3.1. In order for the pattern match to be performed with only the former criterion, th should be set with adequate value (and additional criterion to narrow index candidates, if necessary) so that only the correct index is detected. With the latter criterion, it is easy to see that the index can be correctly recovered if ℓ_w is large. Note that, under the condition that each bit of PUF output is independent of other bits, the distances for correct and incorrect indexes are expected about $p_e \cdot \ell_w$ and $\ell_w/2$, where p_e is a bit error rate, respectively. Namely, if ℓ_w increases, the gap between these distances also increases and the correct index is detectable. In the next section, we check the relation between ℓ_w and the correctness of the recovered index (key). The combination of the first and second criteria, first finding index candidates with a loose threshold and then detecting the index with the smallest distance, enables not only the recovery of correct index but also the detection of system errors from faults and manipulation attacks.

5. Feasibility Tests

In this section, we test the feasibility of the SC-PMKG scheme by simulations and experiments, respectively.

5.1. Simulation. We first check the feasibility of the SC-PMKG scheme by a simulation with an approximation analysis. In this simulation, we estimate the failure probability in the key reconstruction by changing parameters ℓ_w, n, and p_e where p_e is a bit error rate in the PUF output. Following Paral and Devadas [9], for each ℓ_w, we set n so that the entropy of input for KDF is 160-bit. For example, if we use $\ell_w = 32$, we set $n = 32$ as a minimum integer so that $n \log_2 \ell_w \geq 160$.

Table 1 summarizes the failure probability for $p_e \in \{0.15, 0.035\}$ and ℓ_w from 32 to 160. Here, we assume 0.15 and 0.035 for p_e because $p_e = 0.15$ is a well discussed parameter for PUF instantiations and $p_e = 0.035$ is an average of bit error rate in our experiments below. As for n, we set n for each ℓ_w by a minimum integer as in the above case for $\ell_w = 32$. With ℓ_w and n, $\ell_p = \ell_w \times n$ is a length of the PUF output. For each p_e and ℓ_w, the "prob. (sim)" shows the failure probability from the simulation, which is estimated as shown in Algorithm 1.

For each ℓ_w, we tried $N = 5,000,000$ reconstructions, in total. (In Table 1, "0" means that there is no error for $N = 5,000,000$, which means that prob. (sim) is less that $1/N = 2 \times 10^{-7}$.) From our simulation, $\ell_w = 160$ ($p_e = 0.15$) or $\ell_w = 48$ ($p_e = 0.15$) should be enough for the key reconstruction with the failure probability less than 10^{-6}.

We also estimate the failure probability by an approximation analysis, as shown in "prob. (approx)" in Table 1.

Let us discuss the distance between a substring stored in the NVM and one circularly shifted by the *correct* index. The distance follows the binomial distribution $B(\ell_w, p_e)$. We approximate it with the normal distribution $N(\ell_w p_e, \ell_w p_e(1-p_e))$. Similarly, we approximate the distribution of the distance related to an *incorrect* index with the normal distribution $N(\ell_w/2, \ell_w/4)$.

In case the distance for the *correct* index is more than one for the *incorrect* index, the wrong index is recovered. Let us denote the probability by q. The difference between the distances for correct and incorrect indexes follows the normal distribution $N(mean, var)$ where $mean = \ell_w p_e - \ell_w/2$ and $var = \ell_w p_e(1-p_e) + \ell_w/4$, which is a composite (difference) of two normal distributions. With this notation, q can be estimated by

$$q \approx \int_{mean/\sqrt{var}}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left(\frac{x^2}{2}\right) dx. \tag{2}$$

Note that there are $\ell_w - 1$ wrong candidates for one index. If q is small enough, the probability where one of wrong index is recovered is estimated by $1-(1-q)^{\ell_w - 1} \approx 1-\{1-(\ell_w-1)q\} = (\ell_w - 1)q$. Also note that there are n indexes to be recovered. Similarly, the probability where one of index is incorrectly recovered among n substrings is approximated by $(\ell_w - 1)nq$. "prob. (approx)" in Table 1 summarizes the failure probability for each p_e, ℓ_w (and q), and n.

Table 1: Failure probability in SC-PMKG ((a) $p_e = 0.15$, (b) $p_e = 0.035$).

(a)

ℓ_w	n	ℓ_p	prob. (sim)	prob. (approx)
32	32	1024	0.538	0.630
48	29	1392	8.02×10^{-2}	5.40×10^{-2}
64	27	1728	8.29×10^{-3}	4.41×10^{-3}
96	25	2400	9.02×10^{-5}	2.83×10^{-5}
128	23	2944	1.00×10^{-6}	1.69×10^{-7}
160	22	3520	0	1.01×10^{-9}

(b)

ℓ_w	n	ℓ_p	prob. (sim)	prob. (approx)
8	53	424	0.682	0.242
16	40	640	6.86×10^{-2}	7.51×10^{-3}
24	34	816	2.73×10^{-3}	1.45×10^{-4}
32	32	1024	1.29×10^{-4}	2.54×10^{-6}
48	29	1392	2.00×10^{-7}	6.27×10^{-10}
64	27	1728	0	1.36×10^{-13}

Enrollment phase:
(1) Generate an ℓ_p-bit string at random.
(2) Divide the string into n substrings $\{y_i\}_{i\in[1,n]}$ with ℓ_w-bit each.
(3) For $i \in [1, n]$, choose $ind_i \in [0, \ell_w - 1]$ at random.
(4) For $i \in [1, n]$, compute $z_i = \mathsf{rotate}(y_i, ind_i)$.
(5) Store $\{(y_i, ind_i, z_i)\}_{i\in[1,n]}$.
Reconstruction phase:
(1) Set cnt to 0.
(2) Repeat the following steps N times:
(a) For $i \in [1, n]$, generate an ℓ_w-bit string e_i where each bit of e_i is 1 with probability p_e, independently.
(b) For $i \in [1, n]$, set $y'_i = y_i \oplus e_i$ and look for ind'_i where $\mathsf{rotate}(y'_i, ind'_i)$ is near to z_i.
(c) Increment cnt if there exists $i \in [1, n]$ such that $ind_i \neq ind_i$.
(3) Compute the failure probability by cnt/N.

Algorithm 1

Since the approximations are inappropriate for small parameters, there are gaps between the failure probabilities from simulation and our approximation analysis. However, our analysis seems valid for large ℓ_w; namely, $\ell_w = 160$ and $\ell_w = 64$ are large enough, when p_e is 0.15 and 0.035, respectively, so that the failure probability is negligible.

5.2. Experiments with Nucleo Boards. We then test the feasibility with two Nucleo-F401RE boards (http://www.st.com/en/evaluation-tools/nucleo-f401re.html (accessible on Nov. 9, 2018)) by STMicroelectronics. We load $\ell_p = n \cdot \ell_w$ initial SRAM bits on each board and regard them as an output of SRAM-PUF. The experiments are performed in the room temperature and the power is supplied by the USB interface. (The characteristics of PUF would change by environments such as temperature and supplied power voltage. Refer to Chapter 4 of [24], for instance. The success rate of the key reconstruction depends on the bit error rate of PUF. We should select adequate parameters, such as ℓ_w and n, with the worst error rate among the different environments.) The averages of bit error rate for 4,096 SRAM bits on these boards are 3.30% and 3.57%, respectively. Here, the bit error rate is evaluated, in a simple manner, by a difference between the bit for the first read and bits for the following 10,000 reads.

Similar to the above simulation, we estimate the failure probability by changing ℓ_w. For each board, we try the reconstruction 10,000 times. Table 2 summarizes the failure probabilities for two boards, respectively. (Similar to Table 1, "0" means that there is no error for $N = 10,000$ and that the failure probability is less than 10^{-4}.)

For small ℓ_w, the probabilities differ from those of Table 1. This is because, different from the simulation, the initial values of SRAM cells are not uniformly random, and their stabilities are not constant for each bit. With larger ℓ_w, the probabilities in our experiment are close to those in the simulation. Note that since the initial values of SRAM cells are not uniform, if we use this SRAM as SRAM-PUF directly, the resulting SC-PMKG scheme might be insecure. From our observations, the initial values of SRAM cells have certain

Table 2: Failure probability in SC-PMKG with Nucleo-F401RE boards.

ℓ_w	n	ℓ_p	board 1	board 2
8	53	424	1	1
16	40	640	0.137	0.434
24	34	816	8.00×10^{-4}	4.20×10^{-3}
32	32	1024	1.00×10^{-4}	1.00×10^{-4}
48	29	1392	0	0
64	27	1728	0	0

bit patterns; and therefore regarding either a concatenation of first bits of each byte or an XORed value of different cells like 4-XOR PUF as an output of the SRAM-PUF may be a solution to enhance the security of the SC-PMKG scheme.

5.3. Experiments with Open Dataset. In order to check the feasibility with a different type of PUF, we use the ring oscillators' (ROs') frequencies which are available at the Secure Embedded Systems (SES) Lab at Virginia Tech (http://rijndael.ece.vt.edu/puf/download.html (accessible on Nov. 9, 2018)). We select "Full Standard Dataset Oscillator Counts." The dataset includes 100 oscillator counts for 512 ROs of 193 Xilinx Spartan-3E FPGA boards (XC3S500E) under the standard temperature/voltage. Using these ROs' counts, up to $\binom{512}{2} = 130816$ bits of RO-PUF can be derived. We implement the SC-PMKG with some of them. The average bit error rate for the first 1728 RO-PUF bits is 1.21%.

In the dataset, there are only 100 samples for each board. In the feasibility test, we evaluate the average and maximum of failure probabilities over 193 boards. Table 3 summarizes them. (Similar to Table 2, "0" means that the failure probability is less than $1/(99 \times 193) < 5.23 \times 10^{-5}$.)

Since the bit error rate (1.21%) is smaller than that of Nucleo boards (approx. 3%), the failure probabilities are also small. That is, for the correctness, fewer PUF output bits are required, for example, $\ell_p = 1024$.

6. Discussion

Table 4 summarizes the comparison among the PMKG, C-PMKG, and SC-PMKG schemes. Let us discuss the detail of each item below. We refer to [9, 25] for parameters of the PMKG and C-PMKG schemes, respectively. As for C-PMKG, we also add parameters, in parentheses, which we estimate by the similar theoretical approximation of Section 5.1. Note that the authors of [9, 25] decided the parameters so that the failure in key reconstruction should be negligible (with probability less that 10^{-6}).

"|Key|" is a bit length of an output of KGF. References [9, 25] set it to 128, and we follow them for the SC-PMKG scheme. In order to generate the 128-bit key, the previous works [9, 25] assumed that the source of the key had the 160-bit entropy in preparation for the entropy loss of PUF output and the randomness of indexes; i.e., the sum of index lengths was supposed to be 160-bit. We also follow them for the SC-PMKG scheme and give it in the second line "H(Source)" as the entropy of the source.

"H(Index)" shows the entropy of an index. In the PMKG scheme, the index is an address of the first bit of substring. Paral and Devadas [9] set it with 10. In the C-PMKG and SC-PMKG schemes, on the other hand, the index is the amount of circular shift which is at most the length of the substring. Reference [25] followed [9] to set the length of the substring with 256, and therefore H(Index) was set with 8. On the SC-PMKG scheme, from our simulation and experiments, the length of substring seems enough with 160 and we set H(Index) with $\log_2 160$. As in parentheses, the length of substring for the C-PMKG scheme can be similarly estimated with our approximation estimation, so that the failure probability is less than 10^{-6}.

"♯rounds" is a number of rounds. On the PMKG scheme, to generate (ℓ_k =)128-bit key with (ℓ_i =)10-bit index per round, 13 rounds may be enough. However, Paral and Devadas [9] used 16 to make the entropy of source 160-bit. Following them, we set the number of rounds to make the entropy of source 160-bit or more. Similarly, in the C-PMKG scheme, since the length of each index is eight, it is $160/8 = 20$ (or $\lceil 160/\log_2 160 \rceil = 22$ with our approximation). As for the SC-PMKG scheme, the number of rounds is one.

"♯indexes/rounds" are a number of indexes per a round. In the PMKG and C-PMKG schemes, they are one. On the other hand, in the SC-PMKG scheme, it is $\lceil 160/\log_2 160 \rceil = 22$ to achieve the 160-bit entropy.

"|Pattern|" is a bit length of a (sub)string of a PUF output to be pattern-matched. It is the same as the bit length of each auxiliary data. They set it to 256 which is expected to avoid the mismatch. As for the SC-PMKG scheme, we set it to 160 from our simulation. As in parentheses, the length of |Pattern| (and ones of items below) for the C-PMKG scheme can be similarly estimated as ours, with the approximation estimation.

"|PUF output|/round" is a bit length of PUF output in each round. They set it with $1279 (= 1024 + 256 - 1)$ and 256, respectively. As for the SC-PMKG scheme, since it requires $22 (= n)$ substrings of 160-bit in a round as we discuss, the length is $22 \times 160 = 3520$.

"|Total PUF output|" is a bit length of PUF output in total, which is $\ell_p \times r \times n$.

"|Storage|" means the total length of stored data in the NVM, except ind_0. The total length of auxiliary data is estimated with $\ell_w \cdot r \cdot n$. In the SC-PMKG scheme, beside

Table 3: Failure probability in SC-PMKG with ring oscillator (open dataset).

ℓ_w	n	ℓ_p	prob. (ave)	prob. (max)
8	53	424	0.592	1
16	40	640	0.461	1
24	34	816	0	0
32	32	1024	0	0

Table 4: Comparison of PMKG schemes.

Item	PMKG [9]	C-PMKG [25]	SC-PMKG
|Key|, ℓ_k	128	128	128
H(Source)	160	160	160
H(Index)	10	8 ($\log_2 160$)	$\log_2 160$
♯rounds, r	16	20 (22)	1
♯indexes/round, n	1	1	22
|Pattern|, l_w	256	256 (160)	160
|PUF output|/round, ℓ_p	1279	256 (160)	3520
|Total PUF output|	20464	5120 (3520)	3520
|Storage|	4096	5120 (3520)	3776
Additional functions over PMKG	–	rotate	rotate, hash
CS design	Complicated	Complicated	Simple
Snake attacks	Vulnerable	Secure	Secure
Provable security	Unknown	Unknown	Yes

the auxiliary data, the check string is stored in the NVM. We assume that SHA-256 is used as hash and the length is 3520 + 256 = 3776.

Although the C-PMKG and SC-PMKG schemes require fewer PUF outputs in total, they require additional functions, which derive an implementation and computation overhead, compared to the PMKG scheme. In the PMKG scheme, the index search can be performed sequentially. On the other hand, the C-PMKG and SC-PMKG schemes require the additional function rotate. In addition, the SC-PMKG scheme requires the hash function to compute the check string.

The challenge sequencers CS for the PMKG and C-PMKG schemes are carefully designed to avoid an intersection of auxiliary data over rounds as we discussed; however, its design for the SC-PMKG scheme can be simple.

As for the security, the PMKG scheme is vulnerable against the snake attacks. The C-PMKG and SC-PMKG schemes are secure against the snake attacks. Let us discuss the provable security. Although [20] discussed the provable security of the C-PMKG scheme, the security model did not take the manipulation attacks into consideration. It remains an open problem whether the C-PMKG scheme is provably secure without a hash function which, in the SC-PMKG, plays an important role in the provable security. On the other hand, the SC-PMKG scheme is provably secure as in Theorem 5.

To summarize, compared to the PMKG scheme, the C-PMKG and SC-PMKG schemes are not only secure but also efficient; namely, they require less PUF output. In addition, the simple CS design to generate a challenge in the single-round of the SC-PMKG scheme allows us to use even a weak PUF, like the SRAM-PUF, as a building block. It is an advantage for applications with tiny devices, such as the IoT systems.

7. Conclusions

This paper proposes a secure and efficient PMKG scheme, the SC-PMKG scheme, which saves the PUF output and allows a simple construction. It can be realized with even a weak PUF, such as the SRAM-PUF, and is suitable for tiny devices in the IoT systems. Feasibility test with other types of PUF changing the environments is one of our future works.

Appendix

A. Proof of Theorem 5

Proof sketch: We construct $\mathscr{B}$ as follows.

(1) $\mathscr{B}$ randomly generates $ind_0, ind_1, \ldots, ind_n$, where ind_0 is with prefixed length and $ind_i \in [0, \ell_w - 1]$ for $i \in [1, n]$.

(2) $\mathscr{B}$ computes an input $x = \mathsf{CS}(ind_0)$.

(3) $\mathscr{B}$ outputs x as a challenge to receive $y \in \{0,1\}^{\ell_p}$ as a target. If $\mathscr{B}$ is in the real game (the real game is a challenge to $\mathscr{B}$ whether $\mathscr{B}$ can guess that the key is correctly generated; it corresponds to $p_1 = \Pr[\mathscr{A}(y) = 1 \mid y \longleftarrow \mathsf{PBKG}(1^{\ell_k})]$ in Definition 4), y is an output $\mathsf{PUF}(x)$; otherwise, if $\mathscr{B}$ is in the random game (the random game is a challenge to $\mathscr{B}$ whether

$\mathscr{B}$ fails to guess that the key is randomly generated; it corresponds to $p_2 = \Pr[\mathscr{A}(y) = 1 \mid y \longleftarrow \{0,1\}^{\ell_k}]$ in Definition 4), y is a random string over $\{0,1\}^{\ell_P}$. The goal of $\mathscr{B}$ is to distinguish these games.

(4) $\mathscr{B}$ proceeds to steps from (3) to (6) of the enrollment phase in Protocol 3, by regarding y as an output of PUF. Note that, in the random oracle model, $cs = \mathsf{hash}(ind_0, \{(z_i, ind_i)\}_{i\in[1,n]}, key)$ is defined by the random oracle through an oracle query.

(5) $\mathscr{B}$ gives $\mathscr{A}$ key as a challenge.

(6) For a query from $\mathscr{A}$ to the random oracle, $\mathscr{B}$ answers it as follows.

  (a) If the query includes $\{ind_i\}_{i\in[1,n]}$ which $\mathscr{B}$ chose at Step (1) above, $\mathscr{B}$ aborts and fails the proof.

  (b) If the query does not include $\{ind_i\}_{i\in[1,n]}$ which $\mathscr{B}$ chose at Step (1) above, $\mathscr{B}$ passes the query/response between $\mathscr{A}$ and the oracle.

(7) For a query from $\mathscr{A}$ to the SC-PMKG scheme as a PBKG scheme, $\mathscr{B}$ answers it as follows.

  (a) If ind_0, $\{z_i\}_{i\in[1,n]}$, or cs are not modified, $\mathscr{B}$ returns key to $\mathscr{A}$.

  (b) If at least one of ind_0, $\{z_i\}_{i\in[1,n]}$, and cs is modified with different values, $\mathscr{B}$ returns nothing to $\mathscr{A}$.

(8) If $\mathscr{A}$ returns $b \in \{0,1\}$, $\mathscr{B}$ returns b.

In the above construction, $\mathscr{B}$ fails the proof at Step (6)(a), with probability $1/\ell_w^n$. Otherwise, if $\mathscr{A}$ succeeds in distinguishing the games for the SC-PBKG scheme, $\mathscr{B}$ also succeeds in distinguishing the games for the PUF. Therefore, we have the theorem.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number JP18H05289. We sincerely thank Dr. Jeroen Delvaux for his invaluable comments.

References

[1] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *Cryptographic Hardware and Embedded Systems - CHES, 2007., Volume 4727 of Lecture Notes in Computer Science*, P. Paillier and I. Verbauwhede, Eds., pp. 63–80, Springer-Verlag, Berlin, Germany, 2007.

[2] D. E. Holcomb, W. P. Burleson, and K. Fu, "Initial SRAM state as a fingerprint and source of true random numbers for RFID tags," in *Proceedings of the Conference on RFID Security 2007, IEEE*, 1210 pages, 2007.

[3] D. E. Holcomb, W. P. Burleson, and K. Fu, "Power-up SRAM state as an identifying fingerprint and source of true random numbers," *Institute of Electrical and Electronics Engineers. Transactions on Computers*, vol. 58, no. 9, pp. 1198–1210, 2009.

[4] B. Gassend, D. Lim, D. Clarke, M. van Dijk, and S. Devadas, "Identification and authentication of integrated circuits," *Concurrency and Computation: Practice and Experience*, vol. 16, no. 11, pp. 1077–1098, 2004.

[5] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. Van Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *Proceedings of the Symposium on VLSI Circuits (VLSI '04)*, pp. 176–179, Honolulu, Hawaii, USA, June 2004.

[6] D. Lim, *Extracting secret keys from integrated circuits [Msc. thesis]*, Institute of Technology (MIT), Massachusetts, Mass, USA, 2004.

[7] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: how to generate strong keys from biometrics and other noisy data," *SIAM Journal on Computing*, vol. 38, no. 1, pp. 97–139, 2008.

[8] C. Herder, L. Ren, M. Van Dijk, M.-D. Yu, and S. Devadas, "Trapdoor Computational Fuzzy Extractors and Stateless Cryptographically-Secure Physical Unclonable Functions," *IEEE Transactions on Dependable and Secure Computing*, vol. 14, no. 1, pp. 65–82, 2017.

[9] Z. Paral and S. Devadas, "Reliable and efficient PUF-based key generation using pattern matching," in *Proceedings of the 2011 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2011*, pp. 128–133, USA, June 2011.

[10] J. Delvaux and I. Verbauwhede, "Attacking PUF-based pattern matching key generators via helper data manipulation," in *Topics in Cryptology – CT-RSA 2014*, vol. 8366 of *Lecture Notes in Computer Science*, pp. 106–131, Springer International Publishing, Cham, 2014.

[11] J. Delvaux and I. Verbauwhede, "Attacking PUF-based pattern matching key generators via helper data manipulation," in *Topics in cryptology—CT-RSA 2014*, vol. 8366 of *Lecture Notes in Comput. Sci.*, pp. 106–131, Springer, Cham, 2014.

[12] M. Bellare and P. Rogaway, "Random oracles are practical," in *Proceedings of the the 1st ACM conference*, pp. 62–73, Fairfax, Virginia, United States, November 1993.

[13] R. Maes and I. Verbauwhede, "A discussion on the properties of. physically unclonable functions," in *Proceedings of the 3rd International Conference on Trust and Trustworthy Computing (TRUST 2010)*, 2010.

[14] A. R. Sadeghi, *Towards Hardware-Intrinsic Security*, 2010.

[15] A. Datta, A. Derek, J. C. Mitchell, and B. Warinschi, "Key exchange protocols: Security definition, proof method and applications," https://eprint.iacr.org/2006/056.pdf.

[16] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *Advances in Cryptology—(CRYPTO '93)*, vol. 773 of *Lecture Notes in Computer Science*, pp. 232–249, Springer, Berlin, Germany, 1994.

[17] M. Bellare, R. Canetti, and H. Krawczyk, "Modular approach to the design and analysis of authentication and key exchange protocols," in *Proceedings of the 1998 30th Annual ACM Symposium on Theory of Computing*, pp. 419–428, May 1998.

[18] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS '10)*, pp. 237–249, Chicago, Ill, USA, October 2010.

[19] Y. Komano, K. Ohta, and K. Sakiyama, "Encryption key generating apparatus and computer program product (Jan. 2017) Original Japanese patent(5,710,460) was filed on Dec. 16, 2011".

[20] Y. Komano, K. Ohta, K. Sakiyama, and M. Iwamoto, "Provably secure pattern matching key generation using PUF," in *Proceedings of the SCIS 2012, The 2012 Symposium on Cryptography and Information Security*, 2012 (Japanese).

[21] C. Helfmeier, C. Boit, D. Nedospasov, and J.-P. Seifert, "Cloning physically unclonable functions," in *Proceedings of the 2013 6th IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2013*, pp. 1–6, USA, June 2013.

[22] D. Nedospasov, J. Seifert, C. Helfmeier, and C. Boit, "Invasive PUF Analysis," in *Proceedings of the 2013 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pp. 30–38, Los Alamitos, CA, USA, August 2013.

[23] H. Mori, T. Nakagawa, Y. Kitahara et al., "An low-energy 8T dual-port SRAM for image processor with selective sourceline drive scheme in 28-nm FD-SOI process technology," in *Proceedings of the 23rd IEEE International Conference on Electronics, Circuits and Systems, ICECS 2016*, pp. 532–535, Monaco, December 2016.

[24] R. Maes, *Physically Unclonable Functions: Constructions, Properties and Applications*, Springer, New York, NY, USA, 2013.

[25] Y. Iwai, T. Fukushima, D. Moriyama et al., "Implementation and evaluation of PUF-based pattern matching key generation using circular shift," in *Proceedings of the SCIS 2013, The 2013 Symposium on Cryptography and Information Security*, 2013.