Security Mechanisms and Security-Aware Mapping for Real-Time Distributed Embedded Systems

Chung-Wei Lin

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2015-183
http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-183.html

August 10, 2015

Acknowledgement

This work was supported in part by the TerraSwarm Research Center, one of six centers supported by the STARnet phase of the Focus Center Research Program (FCRP), a Semiconductor Research Corporation program sponsored by MARCO and DARPA. This work was supported in part by the Multiscale Systems Center (MuSyC). This work was supported in part by the Industrial Cyber-Physical Systems Center.

Security Mechanisms and Security-Aware Mapping for Real-Time Distributed Embedded Systems

by

Chung-Wei Lin

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering — Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Alberto L. Sangiovanni-Vincentelli, Chair
Professor Dorit S. Hochbaum
Associate Professor Sanjit A. Seshia

Summer 2015

Security Mechanisms and Security-Aware Mapping for Real-Time Distributed Embedded Systems

Abstract

Security Mechanisms and Security-Aware Mapping for Real-Time Distributed Embedded Systems

by

Chung-Wei Lin

Doctor of Philosophy in Engineering — Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Alberto L. Sangiovanni-Vincentelli, Chair

Cyber-security attacks can have a critical impact on embedded systems. They may access secret information, cause system malfunction, or even endanger users in extreme circumstances. These attacks become even more threatening as systems are becoming more connected with the surrounding environment, infrastructures, and other systems. These connections provide breeding grounds for attackers to get access to or take control of the systems. Security mechanisms can be designed to protect against attacks and meet security requirements, such as integrity, authenticity, confidentiality, or availability. However, there are many challenges of applying security mechanisms to embedded systems, such as open environments, limited resources, strict timing requirements, and large number of devices. These challenges make it very difficult and sometimes impossible to add security mechanisms after initial design stages without violating other system constraints. It is therefore important to develop a systematic approach to address security at early design stages together with all other design constraints.

We first propose a general security-aware design methodology which considers security together with other design constraints at design stages. The methodology is based on Platform-Based Design [44], where a functional model and an architectural platform are initially captured separately and then brought together through a mapping process. During mapping, the functional model is implemented on the architectural platform, and constraints and objectives are satisfied and optimized, respectively. Our methodology is different from the traditional mapping process because it not only maps functional models to architectural platforms but also explores security mechanism selection and architecture selection.

We then focus on the security issues for automotive systems as they represent many of the common challenges in embedded systems. We study security for in-vehicle communications and present security mechanisms for the Controller Area Network (CAN) protocol, which is a very representative asynchronous protocol and currently the most used in-vehicle communication protocol. Based on the security mechanisms, we propose a Mixed Integer Linear Programming (MILP) formulation and an MILP-based algorithm to explore task allocation,

signal packing, Message Authentication Code (MAC) sharing, and priority assignment and meet both security and safety constraints. Besides the CAN protocol, we also consider a TDMA-based protocol, which is a very representative synchronous protocol and an abstraction of many existing protocols. The time-delayed release of keys [2, 34, 35, 52] is applied as the security mechanism, and an algorithm that combines a simulated annealing approach with a set of efficient optimization heuristics is developed to solve a security-aware mapping problem for TDMA-based systems. Lastly, we apply our methodology to Vehicle-to-Vehicle (V2V) communications with the Dedicated Short-Range Communications (DSRC) technology. We formulate a security-aware optimization problem and propose an efficient algorithm to solve the security-aware optimization problem.

Experimental results show that our approaches can effectively and efficiently explore design spaces and satisfy all design constraints at design stages. They also demonstrate that security must be considered at initial design stages; otherwise, it is too late to add security after initial design stages.

Simple But Not Easy.

Contents

List of Figures

List of Tables

Acknowledgments

I wound like to express my deepest gratitude to my advisor, Professor Alberto Sangiovanni-Vincentelli, for his visionary guidance throughout my years in Berkeley. I would like to thank Professor Edward Lee, Professor Sanjit Seshia, and Professor Dorit Hochbaum for serving in my qualifying exam committee and providing many inspirational suggestions. I would also like to thank Professor Yao-Wen Chang for his support and encouragement in more than 10 years. I would like to express my heartfelt gratitude to Professor Qi Zhu, Professor Haibo Zeng, Professor Marco Di Natale, Professor Linh Thi Xuan Phan, and Professor David Wagner for their assistance in my research. I would like to thank Joseph D'Ambrosio, Paolo Giusto, and Lei Rao for their great help from General Motors. I would also like to thank Professor Kurt Keutzer and Professor Jaijeet Roychowdhury for establishing my Ph.D. career. My thanks also go to Professor Ai-Chun Pang and Professor Jie-Hong Roland Jiang for their encouragement. Special thanks to Professor Anant Sahai for introducing me to an unexpected but memorable journey—teaching CS70.

I would also like to thank the TerraSwarm Research Center, the Multiscale Systems Center (MuSyC), and the Industrial Cyber-Physical Systems Center (iCyPhy) for their support to my research[1].

I would like to thank Alberto Puggelli for growing up together in Berkeley. I appreciate other group members, John Finn, Liangpeng Guo, Antonio Iannopollo, Mehdi Maasoumy, and Pierluigi Nuzzo, for the wonderful time we share together. I would like to thank Calvin Phung, Bowen Zheng, and Armin Wasicek for their help in my research. I would also like to thank Li-Chiang Lin, Wei-Chang Li, Jun-Chau Chien, and Ming-Fu Lin. Special thanks to San Antonio Spurs for accompanying me in my years as a student. I display my warmest appreciation to my parents, my older brother, and my wife, Yen-Ling Lily Huang, for their endless love and support.

[1]This work was supported in part by the TerraSwarm Research Center, one of six centers supported by the STARnet phase of the Focus Center Research Program (FCRP), a Semiconductor Research Corporation program sponsored by MARCO and DARPA. This work was supported in part by the Multiscale Systems Center (MuSyC). This work was supported in part by the Industrial Cyber-Physical Systems Center.

Chapter 1

Introduction

Cyber-security attacks can have a critical impact on embedded systems. They may access secret information, cause system malfunction, or even endanger users in extreme circumstances. Different types of attacks have been identified in automotive systems [5, 12, 24, 25, 26, 36], aircraft systems [6, 43], global positioning systems [8, 55], medical devices [10, 28], and smart grids [23, 29]. These attacks become even more threatening as systems are becoming more connected with the surrounding environment, infrastructures, and other systems. These connections provide breeding grounds for attackers to get access to or take control of the systems.

Security mechanisms can be designed to protect against attacks and meet security requirements, such as integrity, authenticity, confidentiality, or availability. However, there are many challenges of applying security mechanisms to embedded systems, such as open environments (*e.g.*, wireless communication), limited resources (*e.g.*, network bandwidth, computational ability, and power), strict timing requirements, and large number of devices [10, 22, 23, 26, 43]. These challenges make it very difficult and sometimes impossible to add security mechanisms after initial design stages without violating other system constraints. It is therefore important to develop a systematic approach to address security at early design stages together with all other design constraints.

1.1 Related Work

Security has become a pressing issue for automotive electronic systems, as modern vehicles can be attacked from a variety of interfaces, including direct or indirect physical access, short-range wireless access, and long-range wireless channels [5, 36]. One critical threat is compromising one automotive Electronic Control Unit (ECU) through those interfaces [25]. An attacker may then conduct various attacks by getting access to other ECUs and safety-critical components such as brakes and engines through in-vehicle communications. Another critical threat is directly generating a message on a network through diagnostics ports, empty

ports, or wireless networks [52]. ECUs and safety-critical components may thus behave aberrantly.

An overview of in-vehicle security threats and protections was provided by Kleberger *et al.* [24]. For in-vehicle communication, the Controller Area Network (CAN) protocol has been the most attractive protocol for attackers since it is the most widely used one, and there is no direct support for security protection. Hoppe *et al.* [12] showed the weakness of the CAN protocol that may affect the operations of electric window lift, warning lights, and airbag control system of a vehicle. Koscher *et al.* [25] demonstrated that an attacker is able to take over an ECU and executes many functions such as those of body control modules, engine control modules, and electronic brake control modules. Furthermore, denial-of-service attacks are also possible so that inputs from the driver are ignored. Besides the CAN protocol, Rouf *et al.* [36] studied wireless tire pressure monitoring systems and demonstrate that eavesdropping and spoofing are possible for messages sent from a tire pressure sensor. Checkoway *et al.* [5] conducted comprehensive analysis and experiments on the attack surface of an automotive system. Seifert and Obermaisser [45] developed anomalies and failures detection on the gateway, which can secure in-vehicle network from both external and internal attacks. Wolf and Gendrullis [53] presented a vehicular hardware security module that enables a holistic protection to in-vehicle ECUs and their communications. However, with these potential gateway and hardware protections, protections over communication are still desired. This is because the gateway protections may not be able to protect against all threats (especially those within the same network), while an existing ECU may also be compromised.

Based on the security analysis and study above, the traditional security terminology is adapted to the automotive systems, and there are several possible attack scenarios:

- Modification: an unauthorized node changes existing data.
- Fabrication: an unauthorized node generates additional data.
- Interception: an unauthorized node reads data.
- Interruption: data becomes unavailable.

The modification and the fabrication can be generalized as an unauthorized write of data by an ECU, the interception is generalized as an unauthorized read of data by an ECU, and the interruption is generalized as a Denial-of-Service (DoS) attack. Corresponding to these scenarios, there are several important security properties:

- Integrity: data is not changed (written) or generated by an unauthorized ECU.
- Authenticity: a receiver or a sender is who it claims to be.
- Confidentiality: data is not read by an unauthorized ECU.

- Availability: data is available.

Integrity and authenticity are believed to be more important than confidentiality for automotive communications. This is because automotive systems taken control by an attacker may behave aberrantly and thus have immediate danger, while, regarding confidentiality, the moving behavior of automotive systems is mostly observable by an attacker. As as result, most existing security mechanisms [9, 32, 50] focused on authenticating messages with Message Authentication Codes (MACs) for the CAN protocol. Since a frame of the CAN protocol has only 64 bits for the data payload and the automotive CAN bus speed is typically at only 500 *kbps*, these mechanisms try to reduce the communication overhead of the MACs through various approaches.

The security threats to automotive systems are becoming broader and more challenging with the emerging of Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communications. For instance, Wasicek *et al.* [51] demonstrated potential security threats through V2V communications by modeling a platoon of vehicles accelerating or braking simultaneously under adaptive cruise controls. The major standard for V2V and V2I applications is the Dedicated Short-Range Communications (DSRC). At the message sublayer (above the transport layer) in DSRC, SAE J2735 [40] defines standard message types including a Basic Safety Message (BSM) which contains time, position, velocity, direction, size, and other important information of a vehicle. It enables the development of many safety applications such as forward collision avoidance, lane change warning (blind spot warning), and left turn assist [11]. IEEE 1609.2 [20] provides security services at the DSRC middle layers (network layer, transport layer and message sublayer). Message authentication is supported by using the Elliptic Curve Digital Signature Algorithm (ECDSA), which is an asymmetric cryptographic algorithm. When a vehicle intends to send a message, it signs the message with its private key and sends the message with its signature and certificate digest. A vehicle receiving the message then uses the public key corresponding to the private key to verify the message. The generation time of a message and the location of a vehicle are optionally included in a signed message to protect against replay attacks. Message encryption is also supported in DSRC. More details of DSRC were introduced in some previous works [20, 22].

Besides automotive systems, security is also a rising concern for other embedded systems. Aircraft communicates with ground stations, other aircraft, and satellites through global positioning systems, Automatic Dependent Surveillance Broadcast (ADS-B) [37], and Internet-Protocol-Based Aeronautical Telecommunication Network (IP ATN) [13]. These protocols fulfill the modernization of air transportation systems which become safer, more time and fuel efficient, and more convenient, but there are some potential security risks in global positioning systems and these protocols. Sampigethaya *et al.* [43] introduced current and next-generation aircraft communication protocols and provided an overview of the standardization progress. Especially, security was highlighted due to the risk of attacks from "brought-in" devices of passengers and the higher dependence of flight on data communications. However, most aviation standards have not included security considerations. Zeng *et al.* [55] and Gong *et al.* [8] showed spoofing attacks for global positioning systems.

These attacks may lead global positioning systems to be out of synchronization and affect other systems using global positioning systems, such as aircraft systems and smart grids [8]. By observing the dynamic ranges of successful detection rates, a detection mechanism was proposed to protect against these attacks [55]. Among different types of security properties, integrity, authenticity, and availability are more important than confidentiality for control systems of aircraft [6].

Since many of medical devices utilize wireless communications, possible attacks are also pointed out recently. Halperin *et al.* [10] targeted on an implantable cardioverter defibrillator. By reverse-engineering the implantable cardioverter defibrillator, they successfully performed security attacks, including eavesdropping and spoofing, by replaying signals. They proposed three low-power security mechanisms based on RF power harvesting to protect against these attacks. Furthermore, Li *et al.* [28] demonstrated security attacks on a glucose monitoring and insulin delivery system. They successfully performed passive attacks (eavesdropping of the wireless communication) and active attacks (impersonation and control of the system), which can compromise safety and privacy of patients. They also proposed security mechanisms with cryptographic protocols and body-coupled communication to protect against these attacks.

Security issues in smart grids have also been identified, and some protection guidelines have also been provided. Khurana *et al.* [23] gave an overview of security issues of smart grids. They emphasized the communication and device security as well as privacy and introduced the challenges of security management, such as the complexity and scalability of smart grids. McDaniel and McLaughlin [29] also provided an overview of security issues of smart grids. The privacy concern and vulnerability of devices and systems are emphasized again. Khurana *et al.* [23] worked from authentication principles in Internet protocols and discussed potential constraints of smart grids. They presented several design principles and engineering practices that help ensure the correctness and effectiveness of authentication mechanisms. Metke and Ekl [30] presented several security technologies including public key infrastructures and trusted computing for smart grids. Lastly, due to global positioning systems out of synchronization, fault detection, voltage stability monitoring, and event positioning of smart grids may also be affected [8].

Although many security threats and some security mechanisms were introduced in previous work, embedded systems may still suffer from either lack of appropriate security mechanisms or high overhead resulting from security mechanisms. The limited resources and strict constraints of embedded systems make these challenges more difficult as adding security mechanisms after initial design stages may result in infeasible systems. Therefore, we must propose or apply appropriate security mechanisms and address security at early design stages together with all other design constraints.

1.2 Contributions

In this thesis, we propose a general security-aware design methodology to address security together with other design constraints at the design stages for embedded systems. The methodology is based on Platform-Based Design [44], where a functional model and an architectural platform are initially captured separately and then brought together through a mapping process. During mapping, the functional model is implemented on the architectural platform, and constraints and objectives are satisfied and optimized, respectively. Our methodology is different from the traditional mapping process because it not only maps functional models to architectural platforms but also explores security mechanism selection and architecture selection.

We then focus on the security issues for automotive systems as they represent many of the common challenges in embedded systems, such as resource constraints and timing requirements. We study security for in-vehicle communications and propose security mechanisms for the CAN protocol, which is a very representative asynchronous protocol and currently the most used in-vehicle communication protocol. Based on the security mechanisms, we address security during the mapping from functional models to architectural platforms, and security and safety constraints are considered in an integrated formulation. With a flexible key distribution scheme, the security-aware mapping problem is formulated as a Mixed Integer Linear Programming (MILP) problem. Besides the CAN protocol, a Time Division Multiple Access (TDMA) based protocol for in-vehicle communication is also considered, which is a very representative synchronous protocol and an abstraction of many existing protocols such as the FlexRay [7], the Time-Triggered Protocol [42], and the Time-Triggered Ethernet [41]. This kind of protocols is increasingly adopted in various safety-critical systems for more predictable timing behavior. The time-delayed release of keys [2, 34, 35, 52] is applied as the security mechanism, and an algorithm that combines a simulated annealing approach with a set of efficient optimization heuristics is developed to solve the security-aware mapping problem.

We also apply our methodology to V2V communications with the DSRC technology. We formulate a security-aware optimization problem with consideration of both security and safety requirements, and consider the overhead of different settings of the ECDSA. The key decision variables are the sending rates and the authentication rates of BSMs which carry important information for safety applications and thus need security protections, and their sending rates and authentication rates play dominant roles in system performance and security, respectively [1, 22, 27]. We propose an efficient algorithm to solve the security-aware optimization problem without violating design constraints.

1.3 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 introduces the security-aware design methodology. Chapter 3 presents the security mechanisms for the CAN protocol.

The security-aware mapping problems and algorithms for CAN-based and TDMA-based systems are in Chapter 4 and Chapter 5, respectively. Chapter 6 presents the security-aware optimization problem and algorithm for V2V communications. Lastly, Chapter 7 concludes the thesis and points out some future directions.

Chapter 2

Security-Aware Design Methodology

To provide insights and guidelines for security-aware design problems with limited resources and strict constraints, we propose a general security-aware design methodology, which includes three major components: security-aware mapping, security mechanism selection, and architecture selection, to address security at early design stages together with all other design constraints.

2.1 Security-Aware Mapping

Given a functional model, its *security requirements* include two sets. The first one is a set of *security properties* to be fulfilled, such as integrity, authenticity, confidentiality, and availability. The other one is a set of *security constraints* which are usually quantitative constraints. For example, "authenticity" is the security property to be fulfilled, and "the probability that an attacker successfully guesses a Message Authentication Code (MAC) must be smaller than 10^{-10}" is a security constraint. On the other hand, a *security service* of an architecture platform is a security mechanism directly supported by the architecture or a service which can support other security mechanisms. For example, Physical Unclonable Functions (PUFs) [38, 46, 54] are hardware security services fulfilling integrity and authenticity, while a global time is a security service used to protect against replay attacks and control key-releasing time of the time-delayed release of keys [2, 34, 35, 52].

The security-aware mapping is based on Platform-Based Design [44] paradigm. As shown in Figure 2.1, a functional model and an architectural platform are initially captured separately as abstractions of the application space and the architecture space, respectively. Typically, the abstracted functional model includes its models of computation, functional constraints, security properties to be fulfilled, security constraints, etc. The abstracted architectural platform includes its computational resource, communication resource, storage resource, security services, etc. After the abstractions, the functional model and the architectural platform are brought together through a mapping process so that the functional

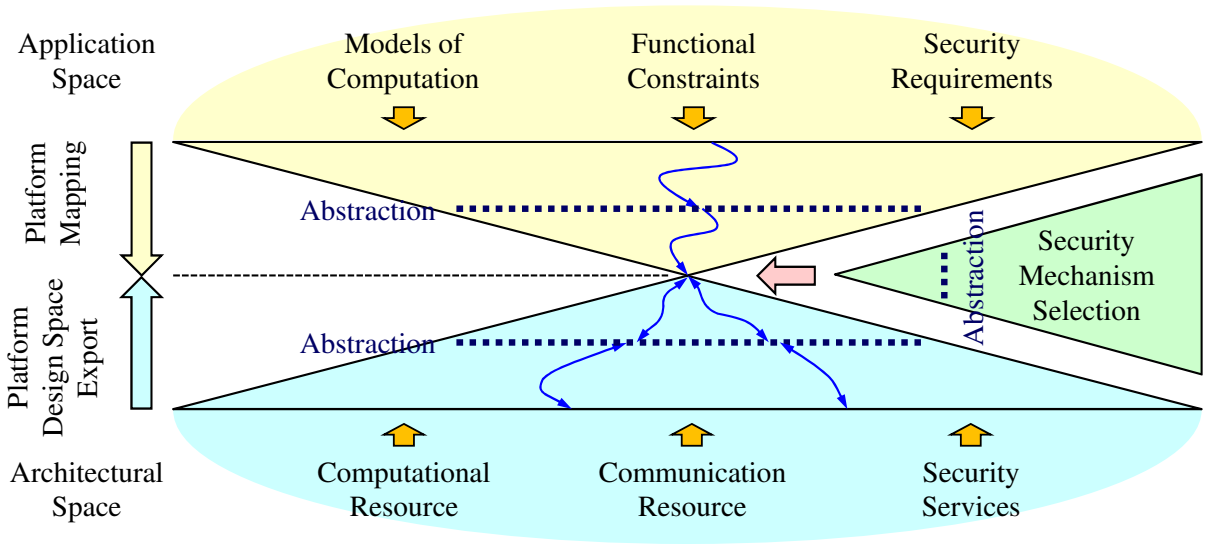


Figure 2.1: The security-aware mapping. A functional model, an architectural platform, and a security mechanism are initially captured separately as abstractions and brought together through a mapping process so that all constraints are satisfied and the design objective is optimized.

model is implemented on the architectural platform while all constraints are satisfied and the design objective is optimized.

Different from the traditional mapping process, the proposed security-aware mapping involves a security mechanism. The security mechanism is also defined through abstractions, including the security properties it supports, the computational, communication and storage overhead, the security constraints, the required security services, etc. During the mapping process, a security property of the functional model must be fulfilled by the security mechanism or the security services of the architectural platform. If the security property is fulfilled by the security mechanism, the required security services of the security mechanism must be fulfilled by the security services of the architectural platform. In addition, there are some security constraints induced by the security mechanism. Security constraints of a security mechanism, along with functional constraints and security constraints of a functional model, must be satisfied through the mapping process.

The mathematical definition of this security-aware mapping problem is defined as follows:

Definition 2.1. A *functional model* is defined by $\mathcal{F} = (M_F, Q_F, R_F, C_F, P_F, O_F)$, where M_F is the models of computation, $Q_F = \{q_1, q_2, \ldots, q_{n_q}\}$ is the set of security properties, $R_F = \{r_1, r_2, \ldots, r_{n_r}\}$ is the set of security constraints, $C_F = \{c_1, c_2, \ldots, c_{n_c}\}$ is the set of other functional constraints, P_F is the set of constant parameters, and O_F is the objective function.

Definition 2.2. An *architectural platform* is defined by $\mathcal{A} = (A_A, V_A, P_A)$, where A_A is the types of architecture, V_A is the set of security services, and P_A is the set of constant parameters.

Definition 2.3. A *security mechanism* is defined by $\mathcal{S} = (S_S, R_S, V_S, P_S)$, where S_S is the types of protection, R_S is the set of security constraints, V_S is the set of required security services, and P_S is the set of constant parameters.

Definition 2.4. $X_{F,A}$ is the set of decision variables of mapping $\mathcal{F}$ on $\mathcal{A}$, and X_S is the set of decision variables of $\mathcal{S}$.

Definition 2.5. The notation $\models$ denotes implementing models of computation M_F, fulfilling a security property $q_i \in Q_F$, satisfying a security constraint $r_i \in R_F \cup R_S$ or a functional constraint $c_i \in C_F$.

Definition 2.6. The *security-aware mapping problem*: given $\mathcal{F}$, $\mathcal{A}$, and $\mathcal{S}$, decide $X_{F,A}$ and X_S such that

$$A_A \models M_F, \tag{2.1}$$

$$\forall q_i \in Q_F, \quad (S_S \models q_i \text{ and } V_A \supseteq V_S) \text{ or } V_A \models q_i, \tag{2.2}$$

$$\forall r_i \in R_F \cup R_S, \quad (X_{F,A}, X_S, P_F, P_A, P_S) \models r_i, \tag{2.3}$$

$$\forall c_i \in C_F, \quad (X_{F,A}, X_S, P_F, P_A, P_S) \models c_i, \tag{2.4}$$

and $O_F(X_{F,A}, X_S, P_F, P_A, P_S)$ is optimized.

Equation (2.2) means that a security property q_i must be fulfilled by the types of protection S_S or the security services V_A. If q_i is fulfilled by S_S, the required security services V_S must be fulfilled by V_A, *i.e.*, V_A supports every required security service to $\mathcal{S}$. Equations (2.3) and (2.4) are usually in a form of $f(X_{F,A}, X_S, P_F, P_A, P_S) \geq 0$, where f is a function.

To solve a security-aware mapping problem, the selection of algorithms depends on the models and the sizes of the problem. In Chapter 4, we use a Mixed Integer Linear Programming (MILP) based algorithm to solve a security-aware mapping problem for Controller Area Network (CAN) based systems. In Chapter 5, we use a simulated annealing based algorithm to solve a security-aware mapping problem for Time Division Multiple Access (TDMA) based systems because the complexity of an MILP-based algorithm is too high for TDMA-based systems. If there is a systematic approach to select algorithms, it can further improve design flows and provide fair comparisons.

2.2 Security Mechanism Selection

A security-aware design problem may not designate a specific security mechanism to be used, and a security mechanism can be selected by system designers to optimize their design objectives, such as system performance, security level, power consumption, or cost.

In this case, given a functional model and an architecture, the goal is to select an appropriate security mechanism from a set of candidates, satisfy all constraints, and optimize the design objective. The security-aware mapping problem with security mechanism selection is defined as follows:

Definition 2.7. The *security-aware mapping problem with security mechanism selection*: given $\mathcal{F}$, $\mathcal{A}$, and a set of security mechanisms $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{n_s}$, decide k $(1 \leq k \leq n_s)$, $X_{F,A}$, and X_{S_k} such that

$$A_A \models M_F, \tag{2.5}$$

$$\forall q_i \in Q_F, \quad (S_{S_k} \models q_i \text{ and } V_A \supseteq V_{S_k}) \text{ or } V_A \models q_i, \tag{2.6}$$

$$\forall r_i \in R_F \cup R_{S_k}, \quad (X_{F,A}, X_{S_k}, P_F, P_A, P_{S_k}) \models r_i, \tag{2.7}$$

$$\forall c_i \in C_F, \quad (X_{F,A}, X_{S_k}, P_F, P_A, P_{S_k}) \models c_i, \tag{2.8}$$

and $O_F(X_{F,A}, X_{S_k}, P_F, P_A, P_{S_k})$ is optimized.

For security mechanism selection, we can solve each corresponding security-aware mapping problem and select the best one. If we can abstract security mechanisms into the same set of variables and parameters, then it is possible to efficiently select a security mechanism in one single security-aware mapping problem.

2.3 Architecture Selection

A security-aware design problem may not designate a specific architecture, either. In this case, given a functional model, the goal is to select an appropriate security mechanism and an appropriate architecture from corresponding sets of candidates, satisfy all constraints, and optimize the design objective. The security-aware mapping problem with security mechanism and architecture selections is defined as follows:

Definition 2.8. The *security-aware mapping problem with security mechanism and architecture selections*: given $\mathcal{F}$, a set of architectures $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_{n_a}$, and a set of security mechanisms $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{n_s}$, decide j $(1 \leq j \leq n_a)$, k $(1 \leq k \leq n_s)$, X_{F,A_j}, and X_{S_k} such that

$$A_{A_j} \models M_F, \tag{2.9}$$

$$\forall q_i \in Q_F, \quad (S_{S_k} \models q_i \text{ and } V_{A_j} \supseteq V_{S_k}) \text{ or } V_{A_j} \models q_i, \tag{2.10}$$

$$\forall r_i \in R_F \cup R_{S_k}, \quad (X_{F,A_j}, X_{S_k}, P_F, P_{A_j}, P_{S_k}) \models r_i, \tag{2.11}$$

$$\forall c_i \in C_F, \quad (X_{F,A_j}, X_{S_k}, P_F, P_{A_j}, P_{S_k}) \models c_i, \tag{2.12}$$

and $O_F(X_{F,A_j}, X_{S_k}, P_F, P_{A_j}, P_{S_k})$ is optimized.

Similar to security mechanism selection, we can solve each corresponding security-aware mapping problem for architecture selection. If we can abstract architectures into the same set of variables and parameters, then it is possible to consider those architectures in one single security-aware mapping problem.

Table 2.1: The security-aware mapping problems for CAN-based systems and TDMA-based systems are interpreted by the proposed methodology. Note that the security properties are different—sharing a secret key between legitimate receivers is allowed for CAN-based systems, but it is not allowed for TDMA-based systems.

		CAN-Based Systems	TDMA-Based Systems
$\mathcal{F}$	M_F	task graph	task graph
	Q_F	authenticity* (message authentication)	authenticity* (message authentication)
	R_F	security risk, MAC length	MAC length
	C_F	path deadline	path deadline
	O_F	latency minimization	latency minimization
$\mathcal{A}$	A_A	distributed nodes, CAN protocol	distributed nodes, TDMA protocol
	V_A	—	global time
$\mathcal{S}$	S_S	shared secret keys (symmetric)	time-delayed release of keys (symmetric)
	R_S	—	key-releasing time
	V_S	—	global time
$X_{F,A}$		task allocation task priority assignment signal packing message priority assignment	task allocation task priority assignment signal packing network scheduling
X_S		receiving group assignment	key-releasing time

2.4 Discussions

In this section, the security-aware mapping problems for CAN-based systems and TDMA-based systems are interpreted by the proposed methodology. We only use them as examples to explain the proposed methodology in this section, and the details of them will be introduced in Chapters 4 and 5, respectively. The two problems are summarized in Table 2.1, where constant parameters P_F, P_A, and P_S are ignored. Note that the security properties are different in the two examples—sharing a secret key between legitimate receivers are allowed for CAN-based systems, but it is not allowed for TDMA-based systems.

The security properties of the two examples are fulfilled by the security mechanisms. As mentioned before, we use an MILP-based algorithm to solve the security-aware mapping problem for CAN-based systems, while we use an SA-based algorithm to solve the security-aware mapping problem for TDMA-based systems.

Authentication mechanisms are used as examples for security mechanism selection. Authentication mechanisms are categorized into four types [52]:

- One-key-for-all key distribution ($\mathcal{S}_1$): the sender and all receivers of a message share and use a symmetric secret key to compute MACs.

- Pair-wise key distribution ($\mathcal{S}_2$): each pair of a sender and a receiver of a message share and use a symmetric secret key to compute MACs.
- Time-delayed release of keys ($\mathcal{S}_3$): the sender of a message uses a symmetric secret key to compute MACs and release the key later.
- Asymmetric cryptography ($\mathcal{S}_4$): the sender of a message uses a private key to sign a message, and a receiver of the message uses the corresponding public key to verify the message.

$\mathcal{S}_1$ and $\mathcal{S}_2$ are two special cases of a flexible key distribution ($\mathcal{S}_5$) for CAN-based systems, when all receivers are in the same receiving group, and all receivers are in their own receiving groups. The examples of security mechanism selection are described as follows:

- For CAN-based systems, $\mathcal{S}_5$ is selected because (1) the security-aware mapping cannot find feasible solutions for $\mathcal{S}_1$ (security risks are too high) and $\mathcal{S}_2$ (MAC lengths are too short), (2) $\mathcal{S}_3$ needs a global time which is not supported by the CAN protocol ($\mathcal{A}$), *i.e.*, (V_A is not a subset of V_{S_3}), and (3) the computational overhead of $\mathcal{S}_4$ is too high.
- For TDMA-based systems, $\mathcal{S}_3$ is selected because the communication or computational overhead of $\mathcal{S}_2$ and $\mathcal{S}_4$ is too high, and two receivers are not allowed to share a secret key, which is possible for $\mathcal{S}_1$ and $\mathcal{S}_5$.

Protections against replay attacks (attackers send messages that they have received from CAN buses or TDMA switches without any modification) are used as examples for architecture selection. A global time can be used to protect against replay attacks. If there is no global time, counters with some synchronization mechanisms can be used. Given an asynchronous protocol ($\mathcal{A}_1$) and a synchronous protocol ($\mathcal{A}_2$), the security-aware mapping can choose to use counters for $\mathcal{A}_1$ or just use the global time supported by $\mathcal{A}_2$. After solving the corresponding mapping problems with security mechanism selection, a better architecture can be selected. The uses of counters and a global time are considered for CAN-based systems and TDMA-based systems, respectively. Note that it is not fair to directly compare the results of CAN-based systems and TDMA-based systems because their security properties are not exactly the same, as mentioned above. If their security properties are the same, we can select the CAN protocol or the TDMA protocol as follows:

- If some legitimate receivers are allowed to share a secret key, both of the results of the CAN protocol (with $\mathcal{S}_5$) and the TDMA protocol (with $\mathcal{S}_3$) are feasible. If the objective is the latency minimization, the CAN protocol has a better objective value; if the objective is the MAC length maximization, the TDMA protocol has a better objective value.
- If two receivers are not allowed to share a secret key, only the result of the TDMA protocol (with $\mathcal{S}_3$) is feasible, so the TDMA protocol should be selected.

Chapter 3

Security Mechanisms for CAN Protocol

As mentioned in Chapter 1, security has become a pressing issue for automotive electronic systems. This is because modern automotive electronics systems are distributed as they are implemented with software running over networked Electronic Control Units (ECU) communicating via serial buses and gateways, but most systems have not been designed with security in mind. This is because the current processes, methods, and tools used for designing current automotive electronics systems focus on safety, reliability, and cost optimization. Although verification for safety and reliability of automotive electronics systems against random failures are commercially available, no security aspect is included as part of the hardware and software architecture development processes, and no standard communication protocol has any built-in provision to prevent or mitigate attacks.

In this chapter, we propose security mechanisms that retrofit the Controller Area Network (CAN) protocol to protect it against masquerade and replay attacks, and the mechanisms can work with the security-aware design methodology in Chapter 2 as candidate security mechanisms for CAN-based systems. We focus on the CAN protocol because it is the most used serial data protocol in current in-vehicle networked architectures and a very representative asynchronous protocol. We address low overhead, high degree of tolerance to faults, and low cost requirements by providing a software-only solution with no additional hardware required.

The security mechanisms are based on message authentication and symmetric secret keys, and counters are also introduced to implement time-stamping of Message Authentication Codes (MACs) in order to overcome the lack of global time in the CAN protocol. We do not focus on the initial security critical key assignment and distribution as this aspect, although very important, is already being studied [47]. Instead, we focus on run-time authentication both in the system steady state (after security secret keys have been distributed to the ECUs) and during running resets experienced by some of the ECUs in systems (when counters are potentially out of synchronization). We also propose two counter reset mechanisms which involve either an ECU that heals itself or a more drastic network-wide counter reset (or re-

synchronization). We provide an analysis of the trade-offs and the benefits versus drawbacks of both approaches. As security has a cost in terms of performance (because of the additional bits needed for MACs and counters) and in terms of potential hazards that may occur due to poor performance, we also work on exploring trade-offs between degree of security and other metrics such as resource utilization. Experimental results show that our security mechanism can achieve sufficient security level without introducing high communication overhead in terms of bus load and message latency.

The chapter is organized as follows. Section 3.1 defines the system model and the attacker model. Section 3.2 presents the security mechanisms. Section 3.3 and Section 3.4 introduce the counter implementation and its corresponding reset mechanisms, respectively. Section 3.5 demonstrates a case study of performance analysis. Section 3.6 provides a summary of this chapter.

3.1 System Model and Attacker Model

For automotive systems and the CAN protocol, integrity and authenticity are very relevant properties which are suitable to our software-only security mechanism solution. To prevent an interruption attack, hardware protections are required as a malicious ECU can freely read data from and write data to a connected CAN bus because of the very same nature of the CAN protocol (broadcast and multi-master with arbitration). Therefore, interruption attacks are outside of the scope of this chapter. We first state our assumptions and provide definitions about the system model as follows:

Assumption 3.1. The network architecture has only one CAN bus, and all ECUs are connected to the bus.

Definition 3.1. A *node* is an ECU.

Definition 3.2. The *sender* of a message is the node sending the message.

Definition 3.3. A *receiver* of a message is a node receiving the message and accepting it by comparing the message ID to the list of its acceptable message ID's.

A sender sends a message by broadcasting it on the CAN bus. Since the CAN protocol is a broadcast protocol, every node "receives" the message, but only receivers (as we have defined them) accept the message.

Assumption 3.2. A node can use volatile (RAM) and/or non-volatile ($FLASH$) memory to store data. Data stored in RAM is no longer available after a node reset; data in FLASH is available after a node reset.

To describe our attacker model, we use a networked architecture topology as in Figure 3.1. Although in the CAN protocol, any node can play the role of sender and receiver in different bus transactions, for illustration purposes, we assume N_1 is a legitimate sender and N_2 is

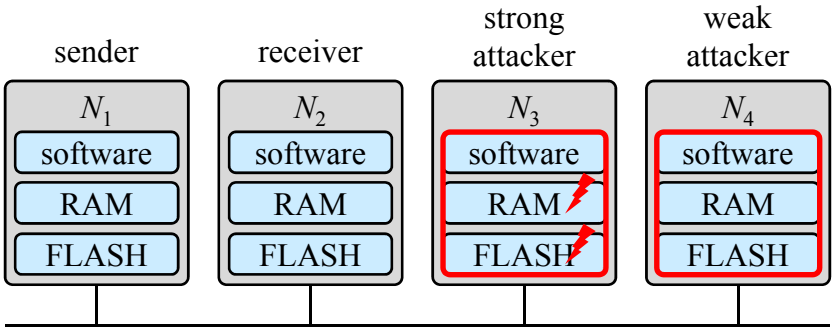


Figure 3.1: The attacker model, where N_1 is a legitimate sender, N_2 is a legitimate receiver, N_3 is a strong attacker, and N_4 is a weak attacker.

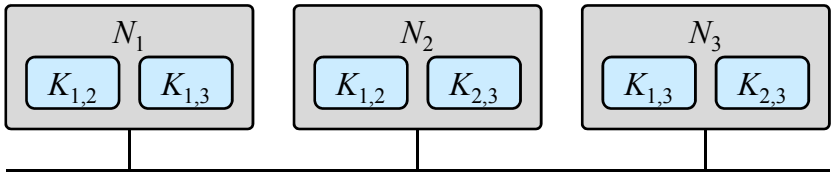


Figure 3.2: The pair-wise secret key distribution for three nodes.

a legitimate receiver. In Figure 3.1, if malicious software takes control of an existing node N_3, it can access any data stored in RAM and FLASH, including data used to implement a security mechanism (*e.g.*, shared secret keys). It is also possible that an attacker uses a node N_4 and connects it to the network (*e.g.*, to perform diagnostics on the network, this node can be a laptop running diagnostic software and connected to the network using the CAN adapter interface); in this case, the malicious software also has access to the RAM and FLASH memory. However, no critical data (*e.g.*, shared secret keys) is stored in RAM and FLASH in the first place.

Definition 3.4. A *strong attacker* is an existing node where malicious software is able to gain control with full access to any critical data.

Definition 3.5. A *weak attacker* is a node where malicious software is able to gain control but no critical data is available (mainly because it was never stored in memory).

Definition 3.6. A *legitimate* node is a node which is neither a strong attacker nor a weak attacker.

For example, in Figure 3.1, N_3 and N_4 are strong and weak attackers, respectively, and N_1 and N_2 are legitimate nodes. The possible attack scenarios that N_3 and N_4 can carry out and that we are addressing with our solution are:

Attack Types	Strong Attacker N_3	Weak Attacker N_4
Modification or Fabrication	Scenario 1	Scenario 2
Replay	Scenario 3	Scenario 4

In the table, we describe the scenario in which a message is supposed to be send by a legitimate sender N_1. However, N_3 and N_4 try to alter this situation. Again, we are not addressing attacks such as a Denial-of-Service (DoS) attack since they will require additional hardware—our proposed solution is software-only. We now explain the scenarios as follows:

- Scenario 1: this is possible if important/secret data between N_1 and N_2 has been stored in RAM or FLASH of N_3. For example, if important/secret data is shared and used by every node in the network[1], then N_3 can use the data stored in RAM or FLASH and pretend to be N_1 to send a new message to N_2 (fabrication).

- Scenario 2: there is no threat because no important/secret data is stored in RAM or FLASH of N_4.

- Scenario 3: this is possible if N_3 reads a message from the CAN bus and then writes the same message to the CAN bus without any modification. Note that, in this case, N_3 does not need to get important/secret data between N_1 and N_2, *e.g.*, a pair-wise secret key as in Figure 3.2, because N_2 will just accept the message by thinking it is sent by N_1.

- Scenario 4: same as Scenario 3.

We now define a masquerade attack and a replay attack as follows [48]:

Definition 3.7. A *masquerade attack* is the scenario that an attacker sends a message in which it claims to be a node other than itself.

Note that a masquerade attack can lead to a fabrication attack, a modification attack, or as a special case, a replay attack:

Definition 3.8. A *replay* attack is the scenario that an attacker sends a message that it has received without any modification.

In the CAN protocol, an attacker performs a reply attack by sending (replaying) the copy of a message that the attacker receives from the CAN bus. The message is not modified or altered, and it is merely sent to other nodes by a node that is not entitled to send it. The other nodes have tables that match the message ID to the sender and thus determine the identity of the sender but have no provision to authenticate it.

Since the CAN protocol is a broadcast protocol, both a strong and weak attacker can successfully carry out a masquerade or replay attack if no security mechanism is put in place.

[1]For example, if the nodes in the network share the same secret key. This is a different scenario from the scenario in Figure 3.2 where nodes share secret keys in a pair-wise fashion.

Table 3.1: The notations in this chapter.

Notation	Explanation
i	the ID of a node.
j	the ID of a node.
k	the ID of a message.
N_i	the node with ID i.
M_k	the message with ID k.
n	the number of nodes.
n_k	the number of receivers of M_k.
$r_{k,s}$	the ID of the s-th receiver of M_k.
f	the function to compute a MAC.
T	the time.
$K_{i,j}$	the shared secret key of N_i and N_j.
$A_{k,s}$	the MAC computed by a sender for the s-th receiver of M_k.
A	the MAC computed by a receiver.
$C_{i,k}$	the counter stored in N_i for M_k .
$C_{i,k}^M$	the most significant bits of $C_{i,k}$.
$C_{i,k}^L$	the least significant bits of $C_{i,k}$.

Even if pair-wise keys are used, a replay attack can still be successful. Before introducing some security mechanisms, we also provide the definition of a *false acceptance* and a *false rejection* as follows:

Definition 3.9. A *false acceptance* is the scenario that a node accepts messages which it should reject.

Definition 3.10. A *false rejection* is the scenario that a node rejects messages which it should accept.

By the definition, a successful attack implies a false acceptance.

3.2 Security Mechanisms

In this section, we will first introduce some basic authentication mechanisms and describe our security mechanisms. In the following sections, we will show the challenges in implementing a security mechanism for the CAN protocol and how we can overcome these difficulties. Some notations that will be used throughout this chapter are defined in Table 3.1.

3.2.1 Basic Authentication

Basic authentication is based on sharing a secret key between a sender N_1 and a receiver N_2 and computing a MAC which is essentially a signature of a message [47]. A key $K_{1,2}$ is the shared secret key stored in N_1 and N_2 and only known by N_1 and N_2. For the sake of the discussion and without loss of generality, we assume a pair-wise secret key assignment as shown in Figure 3.2. N_1 and N_2 perform the following steps to send and receive a message M_k:

Sender (N_1)	
1	$A_{k,1} = f(M_k, K_{1,2})$;
2	Send M_k and $A_{k,1}$;

Receiver (N_2)	
1	Receive M_k and $A_{k,1}$;
2	$A = f(M_k, K_{1,2})$;
3	Accept M_k if and only if $A = A_{k,1}$;

Note that the "1" of $A_{k,1}$ means that N_2 is the first and the only receiver of M_k. Even if N_3 is a strong attacker, since the keys are assigned in a pair-wise fashion, N_3 is not able to compute the MAC (as it is missing $K_{1,2}$) that is needed to attack N_2 with a message that is supposed to be sent by N_1. However, in a broadcast protocol, a message is read by any node in the network, and M_k and $A_{k,1}$ are sent in the clear, so N_3 can read the message and resend it verbatim (essentially replay the same message). Then, N_2 will accept it because the MAC is a match. A possible solution to this problem is to use the concept of global time that allows time-stamping messages. If a global time is adopted then N_2 can prevent a replay attack. An authentication mechanism with global time-stamping is as follows:

Sender (N_1)	
1	Get time T;
2	$A_{k,1} = f(M_k, T, K_{1,2})$;
3	Send M_k and $A_{k,1}$;

Receiver (N_2)	
1	Receive M_k and $A_{k,1}$;
2	Get sending time T;
3	$A = f(M_k, T, K_{1,2})$;
4	Accept M_k if and only if $A = A_{k,1}$;

As in the scenario explained earlier, if N_3 wants to send M_k to N_2, as it cannot retrieve $K_{1,2}$ because it does not have it, it cannot compute the correct MAC. In addition, in case of a replay attack, if N_3 replays the message, it will do so using a MAC with an earlier time stamp that N_2 has used it before. Therefore, the MACs cannot match, and N_2 will reject

the message. As we will show later in this chapter, a global time is not available in the CAN protocol, and thus we introduce monotonic counters to address replay attacks.

The basic authentication mechanisms have been summarized, but there are still other alternatives and variations for authentication. A lot of existing work focus on digital signatures. However, digital signatures have very high communication overhead, making them inapplicable or very difficult to use for the CAN protocol. Szilagyi *et al.* [47, 48, 49] emphasize the constraints in an embedded network and consider a time-triggered (*i.e.*, a global time is available) broadcast protocol. Since every node is a receiver[2], a sent message includes MACs for all receivers. Therefore, N_1 and N_2 perform the following steps to send and receive a message M_k:

Sender (N_i)	
1	Get time T;
2	$\forall j, 1 \leq j \leq n, A_{k,j} = f(M_k, T, K_{i,j})$;
3	Send $M_k, A_{k,1}, A_{k,2}, \ldots, A_{k,n}$;

Receiver (N_j)	
1	Receive $M_k, A_{k,1}, A_{k,2}, \ldots, A_{k,n}$;
2	Get sending time T;
3	Get i where N_i is the sender of M_k;
4	$A = f(M_k, T, K_{i,j})$;
5	Accept M_k if and only if $A = A_{k,j}$;

The authentication operation sends n MACs since the authors use a comprehensive definition of receiver. This means that there are as many receivers as nodes in the network. Each receiver authenticates the message by first identifying the correct MAC that the receiver needs to compare to, based on the information that maps each received message to the unique sender of the message itself. Besides the authentication aspect, the authors have also introduced other interesting features to their authentication mechanism to cope with the potentially limited communication bus data rate and provide fault tolerance. First, only a subset of the MAC bits are sent and used for authentication purposes, *i.e.*, A and $A_{k,j}$ in the above operations are respectively replaced by $[A]_l$ and $[A_{k,j}]_l$ where $[\ ]_l$ is the truncation operation to l bits. Secondly, the analysis assumes that an unsafe state is reached only when some of most recently received messages are successfully attacked. Lastly, in one extension work [49], the authentication is performed by different voting nodes.

Even with the features proposed for reducing the number of transmitted bits and achieving fault tolerance, two major challenges exist in applying the work just described to the CAN protocol. First, the bandwidth available in the CAN protocol is extremely limited. In fact, the maximum and nominal data rate of a CAN bus is only 500 *kbps*, while each standard frame has a maximum total of 134 bits which include 64 bits for payload, 46 bits

[2]The authors use the more comprehensive version of a receiver where a receiver can accept or reject a message.

for overhead (including CRC bits), and 24 bits for bit-stuffing in the worst case [3]. If a security mechanism needs to add MACs to the original frame, as the original frame might have a 64-bit payload, the frame may have to be split into more frames. This may result in increasing bus utilization, a degraded communication performance, or even a unschedulable system. Secondly, as stated earlier, there is no global time in the CAN protocol (a global time is required in those works [47, 48, 49]). To remedy these problems, we propose our security mechanism in the next section.

3.2.2 Our Security Mechanism

The key elements of our proposed security mechanism are stored in each node (in the volatile and non-volatile memory). The elements are: the ID table, the pair-wise symmetric secret keys, and message counters (receiving and sending). In the following, we use our definition of receivers (Definition 3.3).

- ID table: our security mechanism does not use MACs for all nodes [47, 48, 49]. On the contrary, a sender only computes as many MACs as the corresponding receivers of the sent message. This is done by maintaining a ID table in each node where each entry is indexed by a message ID—each entry contains the node ID of the sender and the list of the node ID's of the receivers. We define the ID table with the following function:

$$(i, n_k, r_{k,1}, r_{k,2}, \ldots, r_{k,n_k}) = \text{ID-Table}(k),$$

  where k is the ID of M_k, i is the ID of the sender of M_k, n_k is the number of receivers of M_k, and $r_{k,s}$ is the ID of the s-th receiver of M_k. A sender can check its ID table to determine how many MACs it must compute, what keys it should use, and what ordering of MACs it should attach with the message. A receiver can check the ID table to determine what key it should use and which MAC included in the received frame it should select. Again, the advantage of relying on ID tables is that our mechanism reduces the number of MACs because it considers only the receivers that are accepting the frame after CAN filtering, rather than considering the whole set of receivers that the frame is broadcast to. This can reduce the communication overhead considerably.

- Pair-wise secret key: a pair-wise key $K_{i,j}$ is the "shared secret" between N_i and N_j for authentication. Every pair of nodes has a shared secret key which is not known by any other node. Therefore, any other node cannot modify or fabricate a message, but a replay attack is possible as explained earlier. Note that using pair-wise keys is only a basic key distribution method. If we want to further reduce the communication overhead, we could assign nodes to several groups where each node in a group shares a secret key. Of course, there is a trade-off between security and performance (minimizing communication overhead) in that the security level is diminished but the communication performance is improved.

- Message-based counter: a counter is used to replace the global time and prevent a replay attack. Each node maintains a set of counters, and each counter corresponds to a message, *i.e.*, $C_{i,k}$ is the counter stored in N_i for M_k. If the node is the sender of M_k, its counter value records the number of times that M_k is sent; if the node is the receiver of M_k, its counter value records the number of times M_k has been received (and accepted after being authenticated). Therefore, if a malicious node replays a message, a receiver can check the corresponding receiving counter to see if a message is fresh or not. Because of a network fault, a receiving counter may not have the same value as that of its sending counter. In other words, it is possible that a node sends a frame and updates its sending counter. Then, a network fault occurs, *e.g.*, the electrical bus has a transient fault, and thus the frame never reaches its destination. Therefore, the receiver does not receive the frame and thus does not increase its receiving counter. This means that two counters are out of synchronization. However, our mechanism can deal with this scenario without any loss of security. We will explain this aspect later in this chapter. We now provide the following additional definitions:

Definition 3.11. A *sending counter* for a message is the counter stored in its sender.

Definition 3.12. A *receiving counter* for a message is the counter stored in one of its receiver.

In our security mechanism, every node maintains its ID table, pair-wise keys, and counters. N_i and N_j perform the following steps to send and receive a message M_k:

	Sender (N_i)
1	$(i, n_k, r_{k,1}, r_{k,2}, \ldots, r_{k,n_k}) =$ ID-Table(k);
2	$C_{i,k} = C_{i,k} + 1$;
3	$\forall s, 1 \leq s \leq n_k, A_{k,s} = f(M_k, C_{i,k}, K_{i,r_{k,s}})$;
4	Send $M_k, C_{i,k}, A_{k,1}, A_{k,2}, \ldots, A_{k,n_k}$;

	Receiver (N_j)
1	Receive $M_k, C_{i,k}, A_{k,1}, A_{k,2}, \ldots, A_{k,n_k}$;
2	$(i, n_k, r_{k,1}, r_{k,2}, \ldots, r_{k,n_k}) =$ ID-Table(k);
3	Continue if and only if find $s, 1 \leq s \leq n_k, j = r_{k,s}$;
4	Continue if and only if $C_{i,k} > C_{j,k}$;
5	$A = f(M_k, C_{i,k}, K_{i,j})$;
6	Accept M_k and $C_{j,k} = C_{i,k}$ if and only if $A = A_{k,s}$;

Based on this mechanism, our security mechanism can protect any masquerade attack and replay attack. We claim that by considering the following three scenarios:

- If an attacker sends a message which is not supposed to be received by the receiver, then the receiver will reject the message in Line 3 by checking its ID table.

- If an attacker sends a message which is not supposed to be sent by the attacker, and it is a replay attack, then the receiver will reject the message in Line 4 by checking the counters.
- If an attacker sends a message which is not supposed to be sent by the attacker, and it is not a replay attack, then the receiver will reject the message in Line 6 by comparing the MACs.

3.3 Counter Implementation

The operations in the previous section can meet the requirements stated by our attacker model. However, the number of bits used for the counter must be explored. If the number of bits is not sufficient during the lifetime of a vehicle, then the counter may overflow. For example, if the counter stored at the receiving side overflows and resets to zero, then the replay attack may succeed as the attacker just needs to wait for this event to happen. If the number of bits used for the counter is too large, then the bus will be overloaded.

To remedy these problems, we propose a solution where the counter $C_{i,k}$ is divided into two parts: the most significant bits $C^M_{i,k}$ and the least significant bits $C^L_{i,k}$—only $C^L_{i,k}$ is sent with the message. The steps performed by N_i and N_j are similar, but only $C^L_{i,k}$ is sent. They are listed in Algorithm 1 and Algorithm 2, and the steps performed by a receiver N_j of a message M_k sent by a sender N_i is also shown in Figure 3.3. The two cases are:

- If $C^L_{i,k} > C^L_{j,k}$, then the receiver will use $C^M_{j,k}$ to compute the MAC, which is the same scenario as that of the original mechanism.
- If $C^L_{i,k} \leq C^L_{j,k}$, then the receiver will use $C^M_{j,k} + 1$ to compute the MAC.

If there is a replay attack, the receiver will test $C^L_{i,k} = C^L_{j,k}$ to be true and use $C^M_{j,k} + 1$ to compute the MAC which will be different from the one sent in the replayed message. The receiver will fail the test and reject the message.

The advantage of using this mechanism is that we can reduce the communication overhead without any loss of security. If the receiver consecutively misses several messages due to a network fault, it may reject a message although there is no attack in place because its

Algorithm 1 Algorithm to send M_k.

1: $(i, n_k, r_{k,1}, r_{k,2}, \ldots, r_{k,n_k}) =$ ID-Table(k);
2: $C_{i,k} = C_{i,k} + 1$;
3: **for** each s, $1 \leq s \leq n_k$ **do**
4: $\quad A_{k,s} = f(M_k, C_{i,k}, K_{i,r_{k,s}})$;
5: **end for**
6: Send $M_k, C^L_{i,k}, A_{k,1}, A_{k,2}, \ldots, A_{k,n_k}$;

Algorithm 2 Algorithm to receive and authenticate $(M_k, C^L_{i,k}, A_{k,1}, \ldots, A_{k,n_k})$.

1: $(i, n_k, r_{k,1}, r_{k,2}, \ldots, r_{k,n_k}) =$ ID-Table(k);
2: Find s such that $1 \leq s \leq n_k$ and $j = r_{k,s}$;
3: **if** s is not found **then**
4: Return "Reject";
5: **end if**
6: **if** $C^L_{i,k} > C^L_{j,k}$ **then**
7: $A = f(M_k, C^M_{j,k} | C^L_{i,k}, K_{i,j})$;
8: **if** $A = A_{k,s}$ **then**
9: $C^L_{j,k} = C^L_{i,k}$;
10: Return "Accept";
11: **else**
12: Return "Reject";
13: **end if**
14: **else**
15: $A = f(M_k, (C^M_{j,k} + 1) | C^L_{i,k}, K_{i,j})$;
16: **if** $A = A_{k,s}$ **then**
17: $C^M_{j,k} = C^M_{j,k} + 1$;
18: $C^L_{j,k} = C^L_{i,k}$;
19: Return "Accept";
20: **else**
21: Return "Reject";
22: **end if**
23: **end if**

receiving counter may not be up-to-date (out of synchronization). However, the probability that a counter is out of synchronization is very low. If a counter C is divided into C^M and C^L and the probability of a network fault is q, the probability that a counter is out of synchronization is $q^{2^{|C^L|}}$. For example, if $|C^L| = 3$ and $q = 0.1$, the probability that a counter is out of synchronization is only 0.1^8. Even if this scenario occurs, the MACs will not match, and the receiver will continue rejecting messages (false rejection). Although this scenario is not optimal, a counter out of synchronization is a better option than a successful attack (false acceptance). In addition, we address this potential issue by providing counter reset mechanisms. This is the focus of the next section.

3.4 Counter Reset Mechanisms

A counter reset mechanism is required to deal with an ECU hardware reset or with counters that are out of synchronization because of a network fault. There are two types of hardware resets. The first type is that an ECU may reset as expected, *e.g.*, as it goes

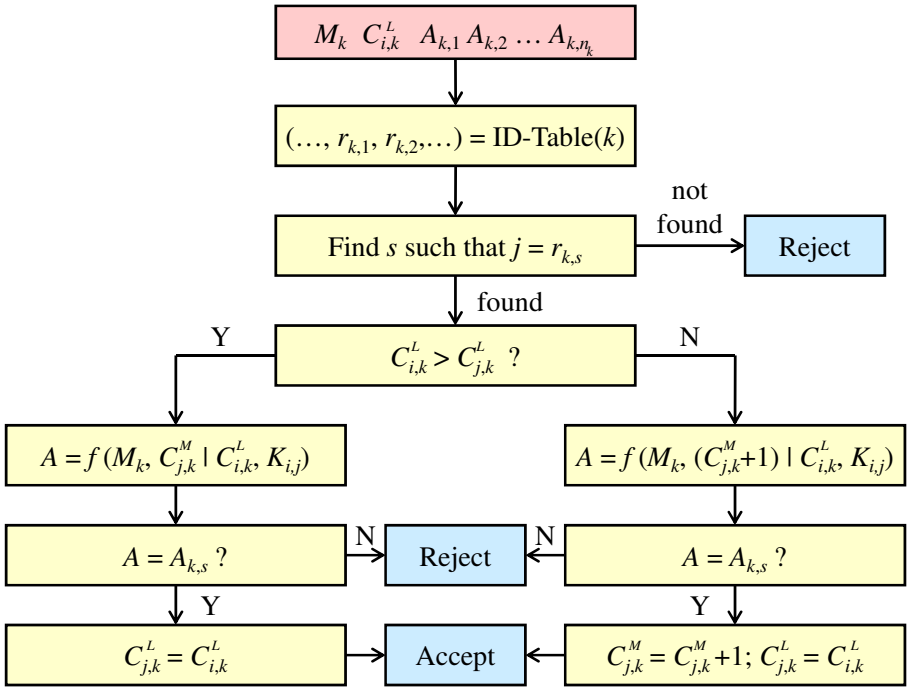


Figure 3.3: The steps performed by a receiver N_j of a message M_k sent by a sender N_i.

into a low power mode as a result of a specific driving mode in which some ECUs are shut off to reduce the energy usage. The other type is that an ECU experiences an unexpected hardware reset due to a power failure. Regardless of the reason why an ECU resets, the rate at which the resets occur or the minimum time interval between them may be too short to allow storing critical data into FLASH which could be restored at a later time. This is because storing data in the FLASH too frequently (at a rate that is higher than of the expected maximum rate of resets) may lead to burning the FLASH itself. Therefore, we devise mechanisms to deal with scenarios where critical data such as updated counter values may not be up-to-date due to hardware resets.

Before an expected shutdown or a change of power state, the ECU copies the relevant data from RAM and stores it into FLASH. At wake-up, the ECU restores the data from FLASH into RAM. However, unexpected shutdowns can occur when hardware failures occur or there is a lack of power, etc. In this case, it is not safe to assume that critical data stored in FLASH can be restored. Therefore, provisions have to be put in place to bring back the ECU, and therefore the system, to a secure state (*e.g.*, with counter values that prevent attacks). Our mechanisms that deal with unexpected hardware resets include "self-healing" and "network-wide" counter resets. The mechanisms provide trade-offs between security levels and communication overhead.

3.4.1 Self-Healing Reset Mechanism

We describe the self-healing reset mechanism performed by a node that has experienced a hardware reset as follows:

- The node sets a FLAG variable to 0.
- The node stores its counters into FLASH every P seconds. The time interval P is a function of the FLASH technology.
- If a node is experiencing an expected hardware reset, then the node tries to store the latest counters value from RAM into FLASH before shutting down. If the operation is successful (it may not be if the FLASH controller refuses to allow it because of potential burning), then FLAG should be set to 1. If not, the remaining steps are the same with those taken for an unexpected hardware reset below.
- If a node reset unexpectedly, nothing can be guaranteed including storing data into FLASH, therefore the FLAG stays at 0.
- When a node wakes up, if FLAG = 1, it restores all counters from FLASH and set FLAG = 0; if FLAG = 0, it restores all counters from FLASH (last counters saved) and increase them by Q, and stores them into FLASH.

P is a parameter that depends on the FLASH technology. There is a trade-off between data freshness and expected life of the FLASH memory. Q is an upper bound of the number of messages that can be sent within the time interval P to prevent a replay attack—different counters can be associated with different values of Q for different messages.

Since the value of Q is an upper bound of the number of messages sent during P, it is possible that this value is not the correct number of messages sent during P. A larger Q value than the real one may lead to false rejections, meaning to a situation where a receiver has a receiving counter that is higher than the sending counter although it should not be. In this case, the receiver will reject a message (even if it should not reject) until the sending counter reaches the value of the receiving counter. Conversely, if Q is smaller than what it should be, then the receiver will accept messages it should not accept (false acceptances). In both cases, the value Q is expected to be tuned off-line. The advantage of this mechanism is that, at wake-up following a node reset, a node resets its counters by itself without the need of additional messages to reset the counters of other nodes. Therefore, the communication overhead is minimized as no network-wide counter synchronization is necessary. However, as the parameter Q is an estimate, potential false rejections or, even worse, false acceptances may occur.

3.4.2 Network-Wide Reset Mechanism

Besides the self-healing reset mechanism, we also propose a network-wide reset mechanism. The key concepts are:

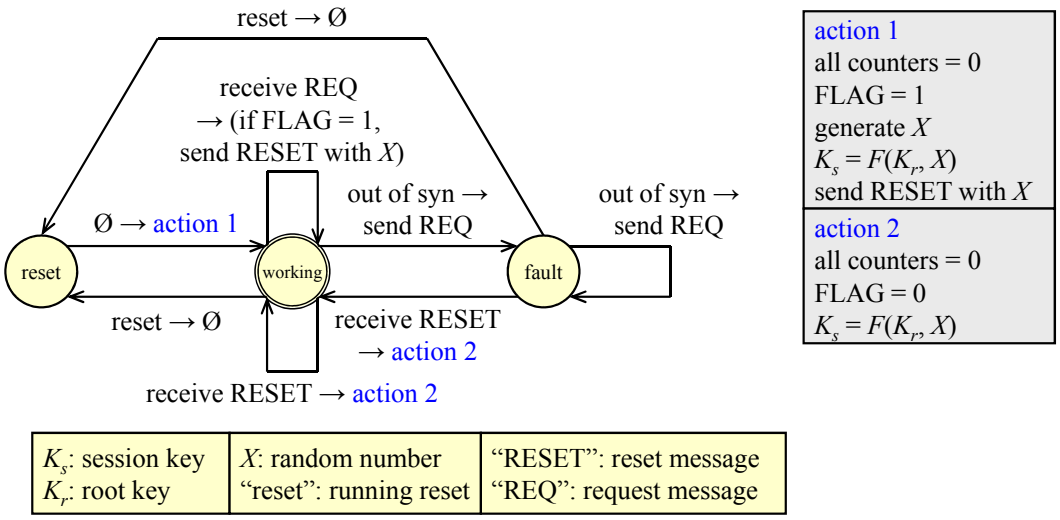


Figure 3.4: The finite state machine of a node in the dynamic network-wide reset.

- A RESET message to set all counters of all nodes to 0.
- A REQ message to achieve fault tolerance.
- New session keys to prevent replay attacks.

In this mechanism, because every counter is reset to 0, new session keys are required; otherwise, an attacker could successfully perform a replay attack. Therefore, a random generated number needs to be included in a RESET message, as it is used to generate the new session key for each node. We can further divide this into two possible approaches. The first one is a "dynamic" network-wide reset where any node experiencing a reset can generate a random number and send a RESET message to all other nodes. The second approach is a "static" network-wide reset where only one special master node can generate a random number and send a RESET message to all other nodes.

The finite state machine of a node in the dynamic network-wide reset is shown in Figure 3.4. This approach has the following features:

- Every node needs to maintain a variable FLAG to indicate if it is the last node generating the random number X and sending the RESET message.
- If a node experiences a reset (goes to the reset state), then it will set all counters to 0, set FLAG to 1, generate a random number X and its new session keys, and send a RESET message with X.
- If a node receives a RESET message, then it will set all counters to 0, set FLAG to 1, and generate its session keys.

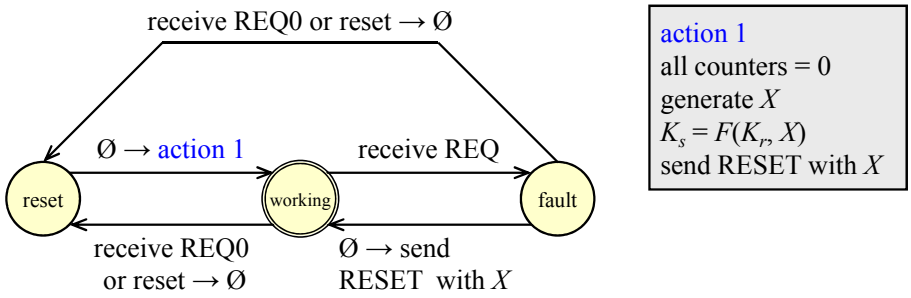


Figure 3.5: The finite state machine of a master node in the static network-wide reset.

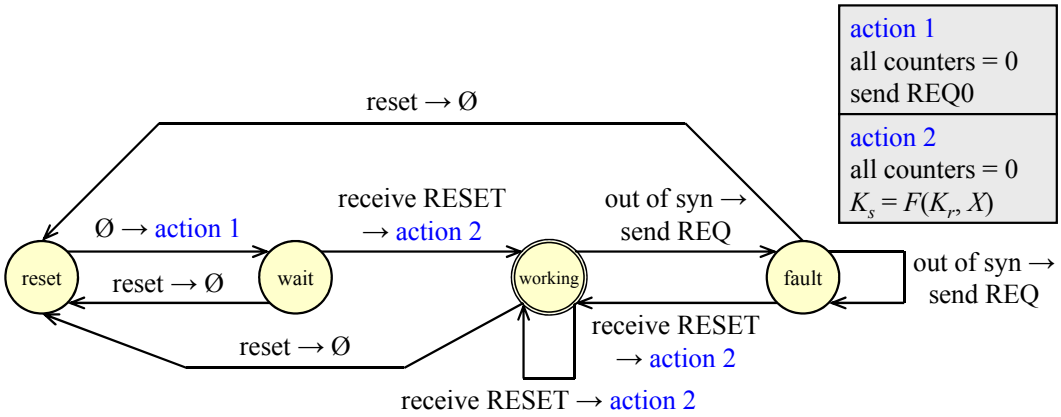


Figure 3.6: The finite state machine of a non-master node in the static network-wide reset.

- If a node finds itself out of synchronization (missing a RESET message due to network fault), then it will send a REQ message to ask for going back to synchronization.
- If a node receives a REQ message, then it will check if FLAG is 1. If yes, it is the last node generating X and sending the RESET message, so it will resend a RESET message.

The finite state machine of a master node in the static network-wide reset is shown in Figure 3.5; the finite state machine of a non-master node in the static network-wide reset is shown in Figure 3.6. The differences between static and dynamic resets are as follows:

- A node does not need to maintain a variable FLAG because only the master node can generate a random number and send a RESET message.
- A REQ0 message is used by a non-master node to ask the master node to reset the network.
- If a non-master node experiences a reset, then it will send a REQ0 message and wait for a RESET message.

- If a master node receives a REQ0 message, it will set all counters to 0, generate a random number X and its session keys, and send a RESET message with X.

Although the network-wide reset mechanism can guarantee that there is no false rejection or successful replay attack, it may determine high transient bus peak loads due to the increasing traffic created by the messages used to reset the counters in every node.

To this point, we have proposed a self-healing and a network-wide (static or dynamic master) reset mechanisms. Both mechanisms provide advantages and disadvantages in terms of security level and bus utilization. In a real case, maybe a mix of them could be applied, depending on the requirements on the communication resource, its available capacity in terms of its data rates, and the secure criticality level of each message.

3.5 Analysis

Our security mechanism has an impact on the system bus load and message latency. To demonstrate them, we formulate the problem as a feasibility analysis problem. The formulation includes the following parameters:

- n: the number of messages.
- q: the probability that a message is missing due to a network fault.
- R: the bus data rate.

The parameters for a message M_k are defined:

- n_k: the number of receivers.
- R_k: the message rate (frequency, as the inverse of its period).
- S_k: the message original size.
- L_k: the upper bound of the total length of the MACs and the least significant bits of the counter.
- C_k: the lower bound of the length of the least significant bits of the counter.
- P_k: the upper bound of the probability of a successful attack.
- Q_k: the upper bound of the probability that a counter is out of synchronization.

If M_k is not a security-critical message, then $C_k = 0$ and $P_k = Q_k = 1$. The following decision variables for M_k are defined:

- x_k: the length of the MAC.
- y_k: the length of the least significant bits of the counter.

The following constraints for M_k are defined:

- The total length of MACs and least significant bits of the counter should be smaller than or equal to L_k.
- The length of least significant bits of the counter should be larger than or equal to C_k.
- The probability of a successful attack should be smaller than or equal to P_k.
- The probability that a node is out of synchronization should be smaller than or equal to Q_k.

The constraints in mathematical forms are defined as follows:

$$\begin{aligned} x_k + y_k &\leq L_k, && (3.1)\\ y_k &\geq C_k, && (3.2)\\ 2^{-x_k} &\leq P_k, && (3.3)\\ q^{2^{y_k}} &\leq Q_k. && (3.4) \end{aligned}$$

The last two constraints also define the probability of a false acceptance (a node accepts messages which it should reject) and a false rejection (a node rejects messages which it should accept). We can easily derive the minimal values of x_k and y_k and then compute the message latency using the equation [31]:

$$l_k = B + \sum_{i \in \mathrm{hp}(k)} \left(\lceil l_k R_i \rceil \frac{S_i + n_i x_i + y_i}{R} \right), \tag{3.5}$$

where l_k is the latency of M_k, $B = \max_i \frac{S_i + n_i x_i + y_i}{R}$, and $\mathrm{hp}(k)$ is the index set of messages with higher priorities than M_k. By using a traditional fix-point calculation, the latency is computed through an iterative method until convergence (if a solution exists).

We use a test case with 17 security-critical messages among 138 messages, and $q = 0.1$, $R = 500$ *kbps*, $L_k = 32$ bits, $C_k = 1$ bit for all security-critical messages. Table 3.2 and Table 3.3 show the relative bus loads and average latencies with different values of P and Q, where $P_k = P$ and $Q_k = Q$ for all k, under the assumptions that the n_k's are 1 and 3, respectively. The number of receivers was not known, so we have used a simple assumption. If this information is provided, more general experiments can be done by assigning different values for P_k and Q_k for different k. Again, the main purpose of this experiment is to show how the security mechanism impacts on the system bus load and message latency. If there exist tight constraints on the bus load, the average message latency, or the message latency (deadline) for each message, then we can check if the security mechanism can be applied or not.

As shown in Table 3.2, when $n_k = 1$, if we want to make sure that the probability of a successful attack and the probability that a node is out of synchronization are both bound

Table 3.2: The relative bus load and average message latency under $n_k = 1$ and different values of P and Q where "—" means that there is no feasible solution. Without the security mechanism, the original bus load 376.44 *kbps* and average message latency 11.535 *ms* are both scaled to 1.

P	Q									
	10^{-1}		10^{-4}		10^{-7}		10^{-10}		10^{-13}	
	Load	Avg L.	Load	Avg L.	Load	Avg L.	Load	Avg L.	Load	Avg L.
10^{-1}	1.0094	1.0241	1.0113	1.0267	1.0131	1.0288	1.0150	1.0322	1.0150	1.0488
10^{-2}	1.0150	1.0322	1.0169	1.0394	1.0188	1.0425	1.0206	1.0445	1.0206	1.0612
10^{-3}	1.0206	1.0445	1.0225	1.0481	1.0244	1.0506	1.0263	1.0571	1.0263	1.0741
10^{-4}	1.0282	1.0591	1.0300	1.0625	1.0319	1.0646	1.0338	1.0668	1.0338	1.0839
10^{-5}	1.0338	1.0668	1.0357	1.0733	1.0375	1.0767	1.0394	1.0789	1.0394	1.0962
10^{-6}	1.0394	1.0789	1.0413	1.0832	1.0432	1.0883	1.0451	1.0968	1.0451	1.1144
10^{-7}	1.0469	1.0987	1.0488	1.1007	1.0507	1.1040	1.0526	1.1061	1.0526	1.1238
10^{-8}	1.0526	1.1061	1.0544	1.1129	1.0563	1.1181	1.0582	1.1213	1.0582	1.1393
10^{-9}	1.0582	1.1213	1.0601	1.1232	—	—	—	—	—	—
10^{-10}	—	—	—	—	—	—	—	—	—	—

Table 3.3: The relative bus load and average message latency under $n_k = 3$ and different values of P and Q where "—" means that there is no feasible solution. Without the security mechanism, the original bus load 376.44 *kbps* and average message latency 11.535 *ms* are both scaled to 1.

P	Q									
	10^{-1}		10^{-4}		10^{-7}		10^{-10}		10^{-13}	
	Load	Avg L.	Load	Avg L.	Load	Avg L.	Load	Avg L.	Load	Avg L.
10^{-1}	1.0244	1.0506	1.0263	1.0571	1.0282	1.0591	1.0300	1.0625	1.0300	1.0795
10^{-2}	1.0413	1.0832	1.0432	1.0883	1.0451	1.0968	1.0469	1.0987	1.0469	1.1164
10^{-3}	1.0582	1.1213	1.0601	1.1232	—	—	—	—	—	—
10^{-4}	—	—	—	—	—	—	—	—	—	—

by 10^{-4}, then there are 3% increase on the bus load and 6.25% increase on the average message latency. Note that, in some cases where the values of P and Q are both large, there is no feasible solution. For our experiments, we show that we can achieve a high security level (*e.g.*, the probability of a successful attack is smaller than 10^{-8}), with a bus load or average message latency increasing less than 6% and 14%, respectively. However, as shown in Table 3.3, when $n_k = 3$, we can see that the feasible region is reduced. This is because it needs 3 MACs, but there are only at most $L_k - C_k$ bits available for them.

3.6 Summary

In this chapter, we describe security mechanisms that retrofit the CAN protocol and protect it against masquerade and replay attacks. Our mechanism is suitable for the CAN protocol because it has a low communication overhead and does not need to maintain a global time. By using counters and only sending the least significant bits of counters, we also propose two counter reset mechanisms in case counters are out of synchronization. Analysis results show that our security mechanism can achieve sufficient security level without introducing high communication overhead in terms of bus load and message latency.

Chapter 4

Security-Aware Mapping for CAN-Based Systems

The mechanisms in Chapter 3 can protect against masquerade and replay attacks for the Controller Area Network (CAN) protocol. However, adding Message Authentication Code (MAC) and counter bits to an existing design may not lead to optimal or even feasible systems because there may not be enough space in messages for the required MAC and counter bits because of the message length limitation (only 64 bits for payload in the CAN protocol [3]). Besides, adding MAC and counter bits increases the message transmission time (in particular if MACs are truncated and transmitted over multiple messages), which may cause the violation of timing constraints and affect system safety. Some extensions of the CAN protocol provide longer message lengths [4, 57]. For instance, the CAN with Flexible Data-Rate (CAN-FD) protocol [4] can allow messages with 64-byte payload. However, the problems above still exist if the MAC and counter bits are added in an ad-hoc fashion or after the other parts of the design are fixed. Therefore, to achieve a secure and safe design, it is crucial to address security together with other objectives such as latency and utilization during the design space exploration of the system.

In this chapter, we propose an integrated Mixed Integer Linear Programming (MILP) formulation to address both the security and the safety requirements during the exploration of the mapping from a functional model to a CAN-based architecture platform. The mapping design space we explore includes the allocation of tasks onto Electronic Control Units (ECUs), the packing of signals into messages, the sharing of MACs among multiple receiving ECUs, and priority assignment of tasks and messages. The security constraints are set to bound security risks. We extend the security mechanism in Chapter 3 by allowing multiple receiving ECUs to share one MAC in a message. This provides more design flexibility under limited resources, while also requires quantitative measurement of the security cost in our formulation. The safety constraints are defined as the end-to-end deadlines for safety-critical functional paths.

To the best of our knowledge, this is the first work to address security and safety in an integrated formulation in the design automation of automotive electronic systems. Based

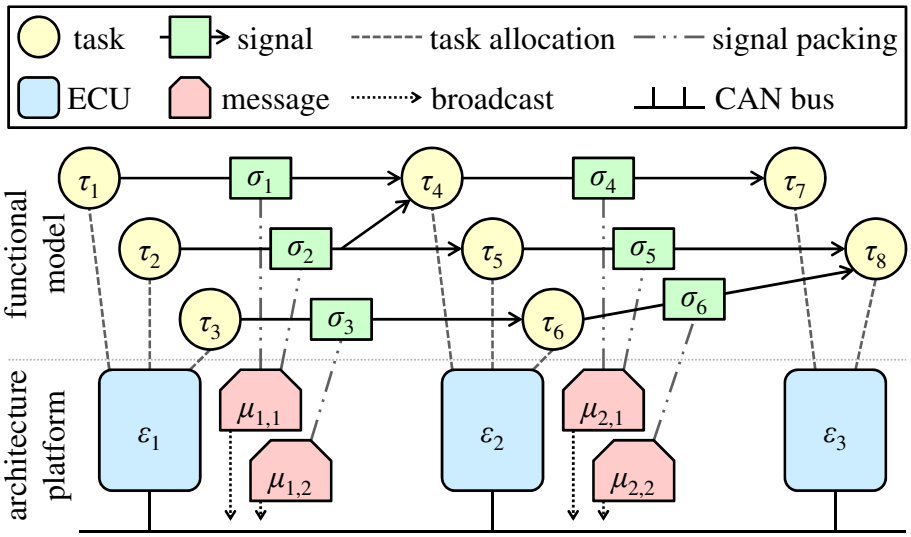


Figure 4.1: The system model of a CAN-based system.

on the optimal MILP formulation, we further propose a three-step algorithm that gradually solves the mapping problem in three simplified MILPs. This approach balances optimality and efficiency and enables solving complex industrial-size problems. We further present an extended formulation which models the path-based security constraints and minimizes security risk directly. Based on the extended formulation, the risk that a functional path being compromised can be bounded and minimized. Along with the extended formulation, we also propose a heuristic algorithm to solve the problem more efficiently. Experimental results of an industrial case study show that our approaches can effectively and efficiently explore the design space to meet the system security and safety requirements.

The chapter is organized as follows. Section 4.1 defines the system model and constraints. Section 4.2 presents the MILP formulation and the three-step MILP-based algorithm. Section 4.3 introduces the extended formulation and algorithm. Section 4.4 demonstrates the experimental results with a case study. Section 4.5 provides a summary of this chapter.

4.1 System Model and Constraints

In this section, we introduce the system model and system constraints involving security and safety.

4.1.1 System Model

The mapping problem addressed in this section is based on Platform-Based Design paradigm [44] and the methodology in Chapter 2, where the functional model and the architecture platform are initially captured separately and then brought together through a

mapping process. As shown in Figure 4.1, the architecture model is a distributed CAN-based platform that consists of $n^{\mathcal{E}}$ ECUs, denoted by $\mathcal{E} = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{n^\varepsilon}\}$, and a CAN bus that connects all the ECUs. Each ECU ε_k can send $n_k^{\mathcal{M}}$ messages, denoted by $\mathcal{M}_k = \{\mu_{k,1}, \mu_{k,2}, \ldots, \mu_{k,n_k^{\mathcal{M}}}\}$. ECUs are assumed to run AUTOSAR/OSEK-compliant operation systems that support preemptive priority-based task scheduling [33]. The bus uses the standard CAN bus arbitration model that features non-preemptive priority-based message scheduling [3]. The functional model is a task graph that consists of $n^{\mathcal{T}}$ tasks, denoted by $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_{n^\tau}\}$, and $n^{\mathcal{S}}$ signals, denoted by $\mathcal{S} = \{\sigma_1, \sigma_2, \ldots, \sigma_{n^s}\}$. Each signal σ_i is between a source task src_{σ_i} and a destination task dst_{σ_i}. Tasks are activated periodically and communicate with each other through signals.

A path π is an ordered interleaving sequence of tasks and signals, defined as $\pi = (\tau_{r_1}, \sigma_{r_1}, \tau_{r_2}, \sigma_{r_2}, \ldots, \sigma_{r_{k-1}}, \tau_{r_k})$. $src(\pi) = \tau_{r_1}$ is the path's source and $snk(\pi) = \tau_{r_k}$ is its sink. Sources are activated by external events, while sinks activate actuators. Multiple paths may exist between each source-sink pair. We assume all tasks in a path perform computations that contribute to a distributed function, from the collection of sensor data to the remote actuations. The worst-case end-to-end latency incurred when traveling a path π is denoted as l_π, which represents the largest possible time interval that is required for the change of the input (or sensed) value at the source to be propagated and cause a value change (or an actuation response) at the sink.

During mapping, the functional model is mapped onto the architecture platform, as shown in Figure 4.1. Specifically, the tasks are allocated onto ECUs, and the signals are packed into messages and transmitted on the CAN bus in a broadcast fashion. Note that messages are triggered periodically and each message contains the latest values of the signals that mapped to it. Static priorities are assigned to tasks and messages for priority-based scheduling at run-time. In addition, the sharing of MACs among receiving ECUs is decided—this is specific to security-aware mapping and will be explained in Section 4.1.2. The design space of task allocation, signal packing, priority assignment, and key sharing is explored with respect to a set of design objectives and constraints. For instance, a path deadline d_π may be set for path π as an application requirement, and we use $\mathcal{P}$ to denote the set of time-sensitive paths with such deadline requirements. There are also utilization constraints on ECUs and the CAN bus, payload size constraints on messages, and constraints on security costs. The details of the security and end-to-end latency constraints are introduced in Section 4.1.2 and Section 4.1.3, respectively, and all constraints are formulated in the MILP formulation in Section 4.2.

4.1.2 Security Constraints and Key Distribution

MACs provide authentication to protect against masquerade and replay attacks, and security constraints should be set at the design time to assure there are enough MAC bits to prevent *indirect attacks* on the MAC bits.

Definition 4.1. An *indirect attack* is the scenario that an attacker does not have the shared secret key between a sender and a receiver so that it can only guess a MAC and attempt to make a message accepted by the receiver.

If there is no prior information and the guess of MAC is purely random, the successful probability of an indirect attack is 2^{-L}, where L is the number of bits of the MAC. In our design formulation, a minimal number of bits is required for each MAC, based on the importance of the signals in the message and the importance of the receivers.

The security mechanism in Chapter 3 uses a dedicated shared key for any pair of nodes, in which case only indirect attacks need to be addressed. However, using such pair-wise key distribution may require a significant number of MAC bits when there are multiple receivers for a message, and may not be feasible. In this chapter, we extend the mechanism in Chapter 3 by defining the notion of *receiving group* to allow multiple receivers to share one MAC in a message, *i.e.*, using the same secret key. This provides more design flexibility but has the risk from *direct attacks*.

Definition 4.2. A *receiving group* of a message is a set of receivers sharing one secret key with the sender of the message.

Definition 4.3. A *direct attack* is the scenario that an attacker gets the shared secret key between a sender and a receiver so that it can pretend as the sender and send a message to the receiver successfully.

In pair-wise key sharing as in Chapter 3, each receiving group contains only one receiver. The example in Figure 4.2(a) shows that one MAC is used for each receiver in the message (6 receivers in total). There is no possibility for direct attack in this case. However, some MACs will not have enough bits available for preventing indirect attack (assuming 32 bits in the message payload are reserved for all MAC bits, then some MACs will have fewer than 6 bits, which means the successful probability of an indirect attack is higher than 3%).

The problem of limited MAC length can be relieved by allowing multiple receivers to share one MAC. A straightforward solution is to use one-key-for-all key distribution, where all receivers are in the same receiving group and use the same key (therefore the same MAC), as illustrated in Figure 4.2(b). This will provide more bits for preventing indirect attack, but it may induce direct attacks—once one ECU in a receiving group is compromised, it can conduct direct attacks on all other ECUs in the same receiving group through masquerade attacks on the message.

In our design formulation, we explore the grouping of receivers into different receiving groups to trade off between direct attack risk and indirect attack risk, based on the total available MAC bits in a message, how critical a message is falsely accepted by a receiver, and how likely an existing node may be compromised. For instance, as illustrated in Figure 4.2(c) and Figure 4.2(d), if N_5 is extremely critical, then no other receiver will be assigned in its receiving group, and there will be no possible direct attack toward it. On the other hand, if N_6 and N_7 are trusted that they are very difficult to be compromised, then they can

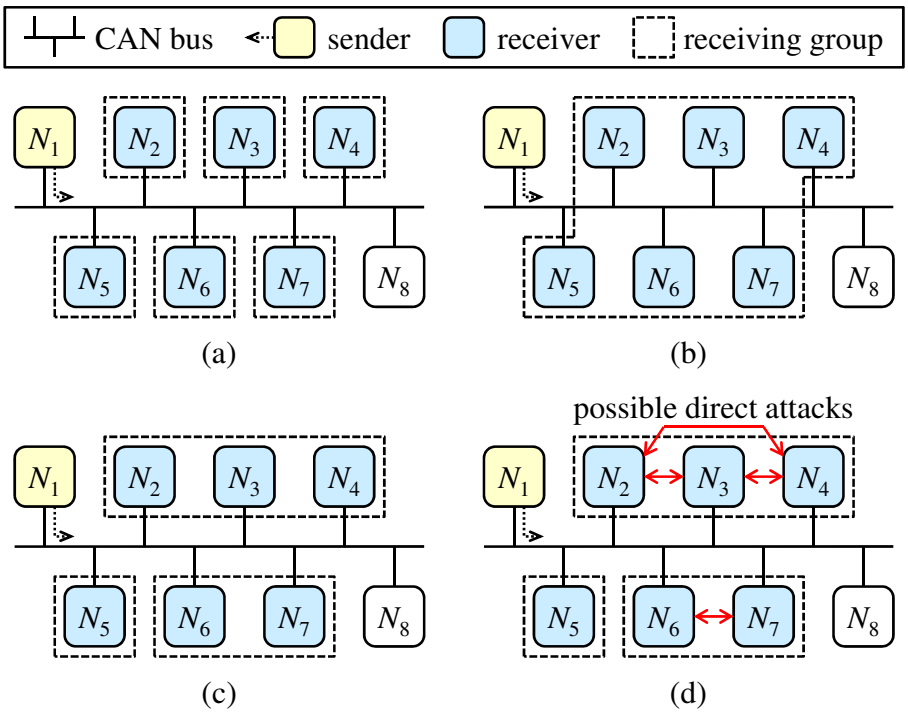


Figure 4.2: Given a message sent by node N_1 and received by N_j ($2 \leq j \leq 7$), (a) the pair-wise key distribution, where 6 MACs are required to be sent with the message, and there is no possible direct attack; (b) the one-key-for-all key distribution, where only 1 MAC is required, but there are possible direct attacks between any pair of receivers; (c) another key distribution, where 3 MACs are required, and (d) there are some possible direct attacks.

be assigned in the same receiving group because the probability of a direct attack between them is very low. We assume the factors that affect direct and indirect attack risks are quantitatively measured and given as parameters in the design inputs, and we set constraints in our formulation to restrict these risks.

4.1.3 Safety Constraints

An important aspect in our approach is to make sure the design with security mechanism still meets the end-to-end latency constraints along functional paths, which directly affect the safety of the system. Assuming an asynchronous sampling communication scheme, the worst-case end-to-end latency of a path π can be computed by adding the worst-case response times of all tasks and global signals on the path, as well as the periods of all global signals and their destination tasks on the path [56]:

$$l_\pi = \sum_{\tau_i \in \pi} r_{\tau_i} + \sum_{\sigma_i \in \pi \wedge \sigma_i \in \mathcal{S}_G} (r_{\sigma_i} + T_{\sigma_i} + T_{dst_{\sigma_i}}), \tag{4.1}$$

Table 4.1: The notations of indices, elements, sets, and quantities.

Notation	Explanation
i, i', j, j'	the index of a task.
k, k', k''	the index of an ECU.
l, l', l''	the index of a message sent from an ECU.
m	the index of a multicast signal from a task.
g	the index of a receiving group of a message.
h	the index of a path.
τ_i	the i-th task.
$\sigma_{i,j}$	a signal between τ_i and τ_j.
ε_k	the k-th ECU.
$\mu_{k,l}$	the l-th message of ε_k.
$\Gamma_{k,l,g}$	the g-th receiving group of $\mu_{k,l}$.
π_h	the h-th path.
$\mathcal{T}$	the set of tasks.
$\mathcal{S}$	the set of signals.
$\mathcal{E}$	the set of ECUs.
$\mathcal{M}$	the set of messages.
$\mathcal{G}_{k,l}$	the set of receiving groups of $\mu_{k,l}$.
$\mathcal{P}$	the set of paths.
$\mathcal{T}^{<}_{i,m}$	the set of receiving tasks of the m-th multicast signal of τ_i.
$n^{\mathcal{T}}$	the number of tasks.
$n^{\mathcal{S}}$	the number of signals.
$n^{\mathcal{E}}$	the number of ECUs.
$n_k^{\mathcal{M}}$	the number of messages of ε_k.
$n_{k,l}^{\mathcal{G}}$	the number of receiving groups of $\mu_{k,l}$.
$n^{\mathcal{P}}$	the number of paths.

where r_{τ_i} and r_{σ_i} are the response times of task τ_i and signal σ_i, respectively, T_{τ_i} and T_{σ_i} are the periods of τ_i and σ_i, respectively, and $\mathcal{S}_G$ is the set of all global signals. The key for calculating end-to-end latency and resource scheduling is to compute the response times of tasks and messages (the response time of a signal is equal to the response time of the message to which the signal is packed into). The task response time of τ_i can be calculated as

$$r_{\tau_i} = C_{\tau_i} + \sum_{j \in \mathcal{T}_H(i)} \left\lceil \frac{r_{\tau_i}}{T_{\tau_j}} \right\rceil C_{\tau_j}, \tag{4.2}$$

where C_{τ_i} is the worst-case execution time of τ_i and $\mathcal{T}_H(i)$ is the set of higher priority tasks on the same ECU of τ_i. The message response time of a message μ_i can be calculated similarly

Table 4.2: The notations of constant parameters.

Notation	Explanation
T_i^τ	the period of τ_i.
$T_{i,j}^\sigma$	the period of $\sigma_{i,j}$.
$T_{k,l}^\mu$	the period of $\mu_{k,l}$.
A	the transmission rate of the CAN bus.
$B_{k,l}$	the blocking time of $\mu_{k,l}$.
$C_{i,k}$	the computation time of τ_i on ε_k.
D_h	the deadline of π_h.
$R_{i,j}$	the maximum allowed security risk of $\sigma_{i,j}$.
$R_{i,j,k',k''}$	the security risk if $\varepsilon_{k'}$ and $\varepsilon_{k''}$ share the corresponding secret key of $\sigma_{i,j}$.
$L_{i,j}$	the data length of $\sigma_{i,j}$.
$L_{k,l,g}$	the reserved MAC length of $\Gamma_{k,l,g}$.
$L'_{i,j,k}$	the required MAC length of $\sigma_{i,j}$ if $\sigma_{i,j}$ is received by ε_k.
M	a large constant for linearization.
H	total length of non-payload part of a message
P	maximum length of payload part of a message

as

$$r_{\mu_i} = C_{\mu_i} + B_{\mu_i} + \sum_{j \in \mathcal{M}_H(i)} \left\lceil \frac{r_{\mu_i} - C_{\mu_i}}{T_{\mu_j}} \right\rceil C_{\mu_j}, \tag{4.3}$$

where C_{μ_i} is the message execution time of μ_i, B_{μ_i} is the blocking time of μ_i, $\mathcal{M}_H(i)$ is the set of higher priority messages of the i-th message, and T_{μ_j} is the period of μ_j. Given a deadline d_π for a path π, the worst-case end-to-end latency l_π of π must be smaller than or equal to d_π.

4.2 Mapping Algorithm

The notations of the indices, elements, sets, and quantities are listed in Table 4.1. We use $\sigma_{i,j}$ to denote a signal from task τ_i to task τ_j (there might be multiple signals between two tasks; we make assure that all of them are considered in our formulation by enumerating $\sigma_{i,j} \in \mathcal{S}$, where S is the entire set of signals). The notations of the constant parameters are listed in Table 4.2, and we assume these parameters are given as design inputs. $R_{i,j}$ is decided by how critical $\sigma_{i,j}$ is. $R_{i,j,k',k''}$ depends on how likely $\varepsilon_{k'}$ may be taken control by a malicious attacker and how much the computation of $\varepsilon_{k''}$ depends on $\sigma_{i,j}$. $L_{i,j}$ includes the payload data length and also the length of its corresponding counter, which is decided in advance by checking the given bound of the probability of a false rejection induced by the mechanism in Chapter 3. $L'_{i,j,k}$ is decided by checking the given bound of the probability of

Table 4.3: The notations of binary variables (their values are 1 if the conditions are true) and real variables.

Notation	Explanation
$a_{i,k}$	τ_i is mapped to ε_k.
$s_{i,j}$	τ_i and τ_j are mapped to the same ECU.
$t_{i,j,k,l}$	$\sigma_{i,j}$ is mapped to $\mu_{k,l}$.
$u_{i,j,k,l}$	$\sigma_{i,j}$ adds its length to $\mu_{k,l}$.
$v_{k,l}$	$\mu_{i,j}$ is non-empty.
$w_{k',k,l}$	$\varepsilon_{k'}$ is a receiver of $\mu_{k,l}$.
$x_{k',k,l,g}$	$\varepsilon_{k'} \in \Gamma_{k,l,g}$.
$y_{k,l,g}$	$\Gamma_{k,l,g}$ is non-empty.
$z_{k',k'',k,l}$	$\varepsilon_{k'}$ and $\varepsilon_{k''}$ are in the same receiving group of $\mu_{k,l}$.
$p_{i,j}$	τ_i has a higher priority than τ_j.
$p_{k,l,k',l'}$	$\mu_{k,l}$ has a higher priority than $\mu_{k',l'}$.
r_i^{τ}	the response time of τ_i.
$r_{k,l}^{\mu}$	the response time of $\mu_{k,l}$.
$r_{i,j}^{\sigma}$	the response time of $\sigma_{i,j}$.
$b_{k,l}$	the total length of $\mu_{k,l}$.
$c_{k,l}$	the computation time of $\mu_{k,l}$.
l_h	the worst-case end-to-end latency of π_h.

a false acceptance induced by the mechanism in Chapter 3. $R_{i,j}$ and $R_{i,j,k',k''}$ address the security risk of a direct attack, and $L'_{i,j,k}$ addresses the security risk of an indirect attack. The current maximum allowed security risk is defined at signal-level, but it can also be defined at receiver-level or at system-level with minor modifications. Finally, the notations of the decision variables are also listed in Table 4.3.

4.2.1 Constraints

In this section, we introduce the various constraints on allocation, security cost, and end-to-end latency. If there is no specific mention, the ranges of variables are $1 \leq i, j \leq n^{\mathcal{T}}$, $1 \leq k \leq n^{\mathcal{E}}$, $1 \leq l \leq n_k^{\mathcal{M}}$, $1 \leq g \leq n_{k,l}^{\mathcal{G}}$, and $1 \leq h \leq n^{\mathcal{P}}$. If a constraint is trivial for all tasks, signals, ECUs, messages, receiving group, or paths, then its "$\forall$" may be omitted.

4.2.1.1 Allocation and Packing Constraints

$$\forall i, \quad \sum_k a_{i,k} = 1; \tag{4.4}$$

$$\forall i, j, k, \quad a_{i,k} + a_{j,k} + s_{i,j} \neq 2. \tag{4.5}$$

Equation (4.4) guarantees that τ_i is allocated to exactly one ECU[1]. Equation (4.5) guarantees that $s_{i,j} = 1$ if and only if there exists k such that $a_{i,k} = a_{j,k} = 1$, satisfying the definition of $s_{i,j}$.

$$\forall \sigma_{i,j} \in \mathcal{S}, k, \quad \sum_l t_{i,j,k,l} = a_{i,k}(1 - a_{j,k}); \tag{4.6}$$

$$\forall \sigma_{i,j} \in \mathcal{S}, k, l, \quad t_{i,j,k,l} \leq v_{k,l}; \tag{4.7}$$

$$\forall \sigma_{i,j} \in \mathcal{S}, k, l, \quad t_{i,j,k,l} T^{\mu}_{k,l} \leq T^{\sigma}_{i,j}; \tag{4.8}$$

$$\forall \sigma_{i,j} \in \mathcal{S}, k, l, \quad t_{i,j,k,l} T^{\sigma}_{i,j} \leq T^{\mu}_{k,l}. \tag{4.9}$$

Equation (4.6) guarantees that $\sigma_{i,j}$ is packed into exactly one message from ε_k, if its source ECU is ε_k and its target ECU is not ε_k. Equation (4.7) guarantees that $v_{k,l} = 1$ if there exists a signal packed into $\mu_{k,l}$. Equations (4.8) and (4.9) guarantee that the period of a signal is equal to the period of the message in which the signal is packed into ($T^{\sigma}_{i,j} = T^{\mu}_{k,l}$ if $t_{i,j,k,l} = 1$).

$$\forall i, k, l, m, \forall \tau_j, \tau_{j'} \in \mathcal{T}^{<}_{i,m}, \quad t_{i,j,k,l} = t_{i,j',k,l}; \tag{4.10}$$

$$\forall i, k, l, m, \forall \tau_j \in \mathcal{T}^{<}_{i,m}, \quad t_{i,j,k,l} = \sum_{\tau_{j'} \in \mathcal{T}^{<}_{i,m}} u_{i,j',k,l}. \tag{4.11}$$

Equation (4.10) guarantees that each branch of a multicast signal is mapped to the same message. Equation (4.11) guarantees that exactly one branch of a multicast signal adds its length to the message.

4.2.1.2 Security Constraints

$$\forall \sigma_{i,j} \in \mathcal{S}, k', k, l, \quad a_{j,k'} + t_{i,j,k,l} - 1 \leq w_{k',k,l}; \tag{4.12}$$

$$\forall k', k, l, \quad \sum_g x_{k',k,l,g} = w_{k',k,l}; \tag{4.13}$$

$$\forall k', k, l, g, \quad x_{k',k,l,g} \leq y_{k,l,g}. \tag{4.14}$$

Equation (4.12) guarantees that $\varepsilon_{k'}$ is a receiver of $\mu_{k,l}$ if there exists a signal $\sigma_{i,j}$ such that τ_j is mapped to $\varepsilon_{k'}$ and $\sigma_{i,j}$ is mapped to $\mu_{k,l}$. Equation (4.13) guarantees that each receiver is in exactly one receiving group. Equation (4.14) guarantees that $y_{k,l,g} = 1$ if there exists a signal mapped to $\mu_{k,l}$ and the signal is in the receiving group $\Gamma_{k,l,g}$.

$$\forall k', k'', k, l, g, \quad x_{k',k,l,g} + x_{k'',k,l,g} + z_{k',k'',k,l} \neq 2. \tag{4.15}$$

[1]In some cases, a task τ_i can only be allocated to a specific ECU ε_k. Then, $a_{i,k}$ should be assigned to 1 directly, and $a_{i,k'}$ is assigned to 0 if $k' \neq k$.

Equation (4.15) guarantees that $z_{k',k'',k,l} = 1$ if and only if there exists g such that $x_{k',k,l,g} = x_{k'',k,l,g} = 1$, satisfying the definition of $z_{k',k'',k,l}$.

$$\forall \sigma_{i,j} \in \mathcal{S}, k, l, \quad \sum_{k',k''} t_{i,j,k,l} \times w_{k',k,l} \times w_{k'',k,l} \times z_{k',k'',k,l} \times R_{i,j,k',k''} \leq R_{i,j}; \quad (4.16)$$

$$\forall \sigma_{i,j} \in \mathcal{S}, k', k, l, g, \quad a_{j,k'} \times t_{i,j,k,l} \times x_{k',k,l,g} \times L'_{i,j,k'} \leq L_{k,l,g}. \quad (4.17)$$

Equation (4.16) guarantees that the security risk (cost) is not larger than the maximum allowed security risk (cost). Equation (4.17) guarantees that the required MAC length is not larger than the reserved MAC length.

Note that the impact of ECUs being compromised is considered in risk parameters $R_{i,j,k',k''}$. As mentioned before, $R_{i,j,k',k''}$ depends on how likely $\varepsilon_{k'}$ may be taken control by a malicious attacker and how much the computation of $\varepsilon_{k''}$ depends on $\sigma_{i,j}$. Such relation can also be modeled *explicitly* by first introducing parameters R_k as the possibility of ε_k being compromised and then modeling $R_{i,j,k',k''}$ as a linear function of R_k and other factors. In this chapter, we focus on addressing the masquerade and replay attacks on security-critical messages and assume $R_{i,j,k',k''}$ are given.

4.2.1.3 End-to-End Latency Constraints

For end-to-end latency constraints, we first model the priority assignment, and then compute the task and message response times, and finally set up the latency constraints on paths.

$$p_{i,j} + p_{j,i} = 1; \quad (4.18)$$

$$p_{i,j} + p_{j,j'} - 1 \leq p_{i,j'}; \quad (4.19)$$

$$p_{k,l,k',l'} + p_{k',l',k,l} = 1; \quad (4.20)$$

$$p_{k,l,k',l'} + p_{k',l',k'',l''} - 1 \leq p_{k,l,k'',l''}. \quad (4.21)$$

Equations (4.18), (4.19), (4.20), and (4.21) guarantee that the priority assignment is feasible.

$$\forall i, \quad r_i^\tau = \sum_k a_{i,k} \times C_{i,k} + \sum_j \sum_k a_{i,k} \times a_{j,k} \times p_{j,i} \times \left\lceil \frac{r_i^\tau}{T_j^\tau} \right\rceil \times C_{j,k}. \quad (4.22)$$

Equation (4.22) computes the task response time of τ_i.

$$\forall k, l, \quad b_{k,l} = H + \sum_{\sigma_{i,j} \in \mathcal{S}} u_{i,j,k,l} L_{i,j} + \sum_g y_{k,l,g} L_{k,l,g}; \quad (4.23)$$

$$\forall k, l, \quad b_{k,l} \leq H + P; \quad (4.24)$$

$$\forall k, l, \quad c_{k,l} = \frac{b_{k,l}}{A}. \quad (4.25)$$

Equation (4.23) computes the total length of $\mu_{k,l}$, taking into account of the data payload, the counter, and the MAC length. Equation (4.24) guarantees that the total message length does not exceed the limit. Equation (4.25) computes the computation time of $\mu_{k,l}$.

$$\forall k, l, \qquad r^{\mu}_{k,l} = B_{k,l} + c_{k,l} + \sum_{k',l'} v_{k',l'} \times p_{k',l',k,l} \times \left\lceil \frac{r^{\mu}_{k,l} - c_{k,l}}{T^{\mu}_{k'l'}} \right\rceil \times c_{k',l'}. \tag{4.26}$$

Equation (4.26) computes the message response time of $\mu_{k,l}$.

$$\forall \sigma_{i,j} \in \mathcal{S}, k, l, \qquad r^{\mu}_{k,l} - M(1 - t_{i,j,k,l}) \leq r^{\sigma}_{i,j}; \tag{4.27}$$

$$\forall \sigma_{i,j} \in \mathcal{S}, k, l, \qquad r^{\sigma}_{i,j} \leq r^{\mu}_{k,l} + M(1 - t_{i,j,k,l}); \tag{4.28}$$

$$\forall \sigma_{i,j} \in \mathcal{S}, \qquad r^{\sigma}_{i,j} \leq M(1 - s_{i,j}). \tag{4.29}$$

Equations (4.27), (4.28), and (4.29) compute the signal response time of $\sigma_{i,j}$. If $\sigma_{i,j}$ is mapped to $\mu_{k,l}$, then $r^{\sigma}_{i,j} = r^{\mu}_{k,l}$; otherwise, if it is not mapped to any message (its source ECU and target ECU are the same), $r^{\sigma}_{i,j} = 0$.

$$\forall h, \qquad \sum_{\tau_i \in \pi_h} r^{\tau}_i + \sum_{\sigma_{i,j} \in \pi_h} (r^{\sigma}_{i,j} + (1 - s_{i,j})(T^{\sigma}_{i,j} + T^{\tau}_j)) \leq D_h. \tag{4.30}$$

Equation (4.30) computes the worst-case end-to-end latency of π_h and guarantees that its deadline is satisfied.

4.2.1.4 Conversion to Linear Constraints

In above formulation of the constraints, there are four cases where the formulation is not linear. They are inequalities of summations of three binary variables (Equations (4.5) and (4.15)), ceiling functions (Equations (4.22) and (4.26)), multiplications of two binary variables (Equations (4.6), (4.16), (4.17), (4.22), and (4.26)), and multiplications of one binary variable and one non-integer variable (Equations (4.22) and (4.26)). The first three cases can be converted into equivalent linear formulations based on their specific representations. The fourth case is more general and can be converted into equivalent linear formulations by introducing a large constant M. The details of the conversions to linear constraints are explained as follows:

- Inequalities of summations of three binary variables in Equations (4.5) and (4.15): if α, β, and γ are binary variables, then we replace the constraint $\alpha + \beta + \gamma \neq 2$ by three constraints:

$$\alpha + \beta - \gamma \leq 1; \tag{4.31}$$

$$\alpha - \beta + \gamma \leq 1; \tag{4.32}$$

$$-\alpha + \beta + \gamma \leq 1. \tag{4.33}$$

- Ceiling functions in Equations (4.22) and (4.26): if α is a function and $\lceil \alpha \rceil$ exists in a constraint, then we replace $\lceil \alpha \rceil$ by an integer variable β and add one constraint:

$$0 \leq \beta - \alpha \leq 1, \tag{4.34}$$

  which is a linear constraint if α is a linear function.

- Multiplications of two binary variables in Equations (4.6), (4.16), (4.17), (4.22), and (4.26): if α and β are binary variables and $\alpha \times \beta$ exists in a constraint, then we replace $\alpha \times \beta$ by a binary variable γ in the constraint and add one constraint:

$$\alpha \times \beta = \gamma. \tag{4.35}$$

  Next, $\alpha \times \beta = \gamma$ can be replaced by equivalent constraints:

$$\alpha + \beta - 1 \leq \gamma; \tag{4.36}$$
$$\gamma \leq \alpha; \tag{4.37}$$
$$\gamma \leq \beta. \tag{4.38}$$

  In fact, if $1 \leq i \leq n$, α_i is a binary variable, and $\prod_{1 \leq i \leq n} \alpha_i$ exists in a constraint, then we can replace $\prod_{1 \leq i \leq n} \alpha_i$ by a binary variable γ in the constraint and add one constraint:

$$\prod_{1 \leq i \leq n} \alpha_i = \gamma. \tag{4.39}$$

  Next, $\prod_{1 \leq i \leq n} \alpha_i = \gamma$ can be replaced by equivalent constraints:

$$\textstyle\sum_{1 \leq i \leq n} \alpha_i - (n-1) \leq \gamma; \tag{4.40}$$
$$\forall i,\ \gamma \leq \alpha_i. \tag{4.41}$$

- Multiplications of one binary variable and one non-integer variable in Equations (4.22) and (4.26): if α is a binary variable, β is a non-integer variable, and $\alpha \times \beta$ exists in a constraint, then we replace $\alpha \times \beta$ by a non-integer variable γ in the constraint and add one constraint:

$$\alpha \times \beta = \gamma. \tag{4.42}$$

  Next, $\alpha \times \beta = \gamma$ can be replaced by equivalent constraints:

$$0 \leq \gamma \leq \beta; \tag{4.43}$$
$$\beta - M(1-\alpha) \leq \gamma \leq M\alpha. \tag{4.44}$$

By applying these rules, the formulation can be transformed to an MILP formulation.

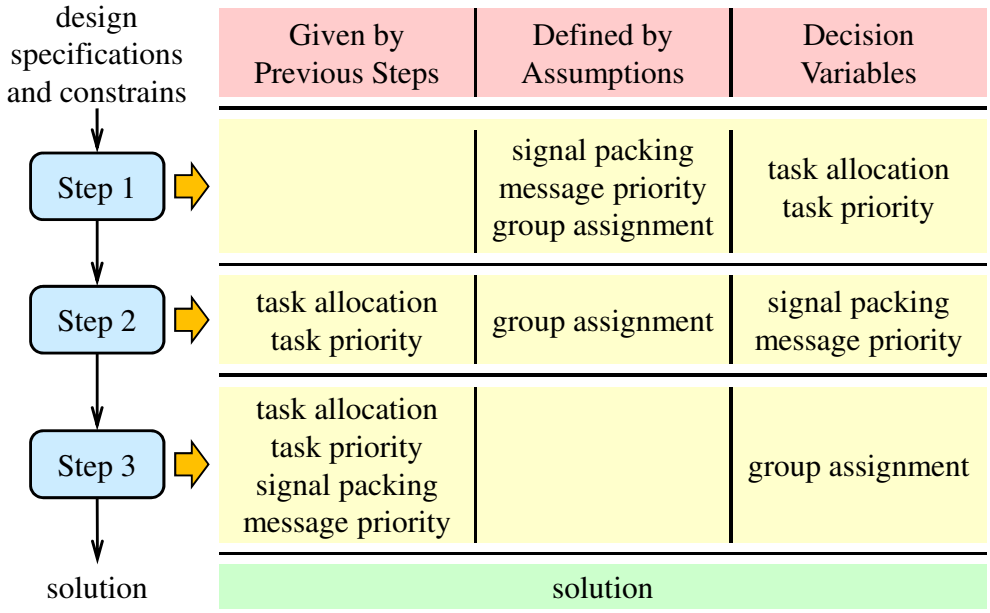


Figure 4.3: The flow of three-step MILP-based algorithm, where "group assignment" means "receiving group assignment." The task allocation and the task priority are solved in Step 1; the signal packing and the message priority are solved in Step 2; the receiving group assignment is solved in Step 3.

4.2.2 Objective Function

The objective function can be defined to minimize the summation of the end-to-end latencies of selected paths:

$$\forall \pi_h, \quad l_h = \sum_{\tau_i \in \pi_h} r_i^\tau + \sum_{\sigma_{i,j} \in \pi_h} (r_{i,j}^\sigma + (1 - s_{i,j})(T_{i,j}^\sigma + T_j^\tau)); \tag{4.45}$$

$$\min \quad \sum_{\pi_h \in \mathcal{P}} l_{\pi_h}, \tag{4.46}$$

where l_h is the worst-case end-to-end latency of π_h and $\mathcal{P}$ is the set of selected paths. The objective function can also be defined to minimize the total security risk with minor modifications.

4.2.3 MILP-Based Algorithm

The MILP formulation introduced above provides an optimal solution but has high complexity. To address complex industrial-size problems, we propose a three-step algorithm, where each step solves part of the mapping problem in a simplified MILP formulation (derived from the original optimal MILP). The flow of the algorithm is shown in Figure 4.3.

- In Step 1, we assume that (1) each message is only reserved for one signal, (2) a MAC with maximum required length for the signal is included in each message, and (3) the priorities of the messages are assigned by the Rate Monotonic policy, *i.e.*, messages with smaller periods have higher priorities. Based on these assumptions, we simplify the MILP formulation introduced above and optimize the task allocation and the task priority.

- In Step 2, having the task allocation and the task priority from Step 1, we optimize the signal packing and the message priority in a simplified MILP formulation, with the assumption that a MAC with maximum required length for the signal is included in each message.

- In Step 3, having the task allocation, the signal packing, and the task and message priorities from previous two steps, we optimize the receiving group assignment. For each message, its length is minimized, and its receiving group assignment satisfies the constraints of security risks (from the perspective of protecting against a direct attack) and required MAC lengths (from the perspective of protecting against an indirect attack).

The formulation of Step 1 is motivated by the observation that task allocation and priority typically have the most significant impact on path latencies and resolving them significantly simplifies the problem for the following steps. The division of Step 2 and Step 3 further reduces the complexity. In addition, if designers are given an existing mapped system and wants to improve the security level by exploring different key sharing strategies, the MILP formulation in Step 3 can be directly applied.

4.3 Extension

In this section, we extend the problem to a new formulation, focusing on new path-based security constraints and a new objective function. Besides them, the rest of the formulation is similar to that in the previous section. We also propose a new efficient mapping algorithm so solve the problem.

4.3.1 Path-Based Security Constraints

There are two types of security constraints in Section 4.2. The first one guarantees that the security risk of direct attacks to a signal is not larger than its maximum allowed security risk, as listed in Equation (4.16). The second type of security constraints bounds the security risk of indirect attacks by assigning enough MAC lengths, as listed in Equation (4.17). The above individual signal-based security constraints do not fully reflect real security requirements. As stated before, security requirements should be defined based on sensor-to-actuator paths. An example is shown in Figure 4.4. Protecting $\mu_{1,1}$ and $\mu_{3,1}$ is not enough because an

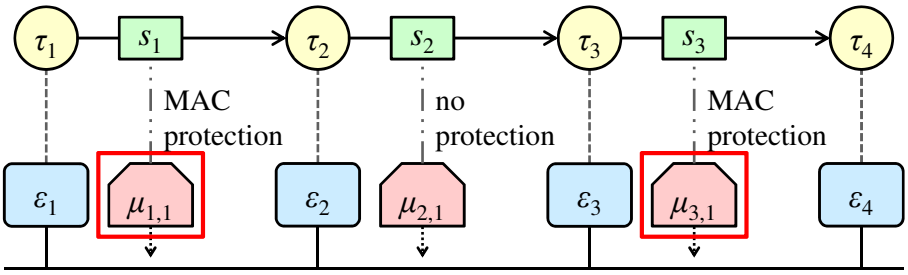


Figure 4.4: Security should be modeled by a path-based security constraint considering possible attacks triggering an actuator. Protecting $\mu_{1,1}$ and $\mu_{3,1}$ is not enough because an attacker can still attack $\mu_{2,1}$ and result in triggering τ_4 on ε_4.

attacker can still attack $\mu_{2,1}$ and result in triggering τ_4 on ε_4. Therefore, in this section, we propose to apply the following constraints. The first one guarantees that the security risk of direct attacks to a path is not larger than its maximum allowed security risk:

$$\forall \pi_h, \qquad \sum_{\sigma_j \in \pi_h} \sum_{\mu_{k,l}} \sum_{\varepsilon_{k'}, \varepsilon_{k''}} x_{j,k,l,k',k''} \times R_{j,k',k''} \leq R_h, \tag{4.47}$$

where R_h is the maximum allowed security risk of π_h. Similarly, R_h is decided by system designers, and it should depend on how critical π_h is. The second type of security constraints guarantees that the MAC lengths of all signals in a path are long enough:

$$\forall \pi_h, \sigma_j \in \pi_h, \varepsilon_{k'}, \mu_{k,l}, \Gamma_{k,l,g}, \qquad y_{j,k',k,l,g} \times L_h \leq L_{k,l,g}, \tag{4.48}$$

where L_h is the required MAC length of any signal in π_h.

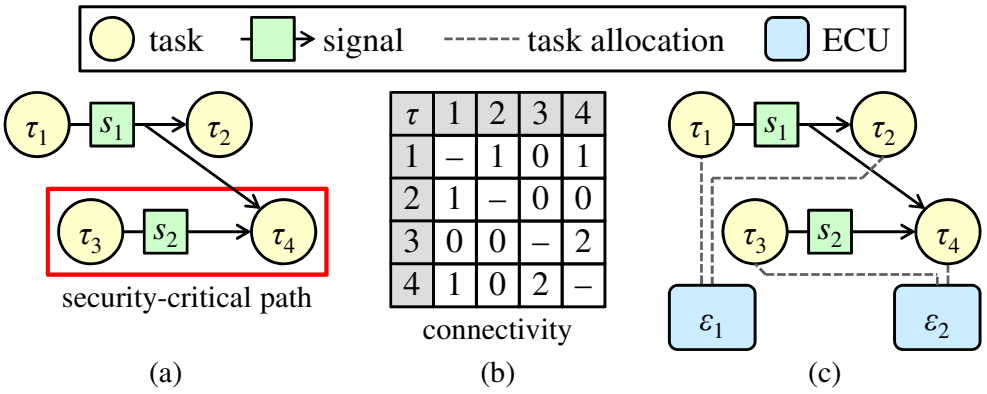
Note that Equation (4.48) can be transformed to Equation (4.17) by assigning appropriate MAC lengths. However, Equation (4.47) cannot be transformed to Equation (4.16) by distributing the maximum allowed security risk of a path into its signals. This is because there may exist forks in a task graph, and a signal may be involved in many paths. As a result, it is very difficult to just reduce the new formulation to the original formulation, and it is necessary to propose the new formulation.

4.3.2 Objective Function

To minimize the security risk of direct attacks, the primary objective function for direct attacks can be defined as:

$$\min \sum_{\pi_h} \left(\sum_{\sigma_j \in \pi_h} \sum_{\mu_{k,l}} \sum_{\varepsilon_{k'}, \varepsilon_{k''}} x_{j,k,l,k',k''} \times R_{j,k',k''} \right). \tag{4.49}$$

If the security risk of direct attacks is minimized, we can also try to increase MAC lengths, which minimizes the security risk of indirect attacks. However, since direct attacks are more



τ	1	2	3	4
1	–	1	0	1
2	1	–	0	0
3	0	0	–	2
4	1	0	2	–

Figure 4.5: (a) Given the task graph, (b) compute the connectivity with both weights being 1 and (c) allocate the tasks to the ECUs.

threatening, we will focus on minimizing the security risk of direct attacks, as shown in Equation (4.49).

4.3.3 Algorithm

We propose an efficient and effective heuristic optimization algorithm. It has been observed that task allocation and priority typically have the most impact on objectives, and solving them significantly simplifies the following steps, so they are the first two steps in the proposed approach.

- Step 1: task allocation. First, the connectivity of each pair of tasks is computed. The connectivity between two tasks is defined as a weighted summation of the number of times that a signal between the two tasks is in a path and the number of times that a signal between the two tasks is in a security-critical path. This definition makes sure that the higher connectivity between two tasks, the more beneficial (in terms of minimizing worst-case end-to-end latency and minimizing security risk) if they are allocated to the same ECU. After computing the connectivity of each pair of tasks, for each ECU, the most connected pair of tasks (among tasks not allocated yet) is selected, and both of them are allocated to the ECU. Then, the most connected task (among tasks not allocated yet) to all tasks allocated to the ECU is selected and allocated to the ECU. We repeat selecting one task and allocating it to the ECU until allocating the task to the ECU makes the utilization of the ECU exceeding a threshold. If allocating a task to the ECU makes the utilization of the ECU exceeding a threshold, we will consider the next ECU. An example is illustrated in Figure 4.5. Given the task graph in Figure 4.5(a), we compute the connectivity of each pair of tasks, as shown in Figure 4.5(b). After that, τ_3 and τ_4 are selected and allocated to ε_2. Then, τ_1 is selected, but we assume that allocating τ_1 to ε_2 makes the utilization of ε_2 exceeding a

threshold. As a result, τ_1 and τ_2 are allocated to ε_1, and the final allocation is shown in Figure 4.5(c).

- Step 2: task priority assignment. The task priority is assigned according to the Rate-Monotonic Policy, *i.e.*, tasks with smaller periods have higher priorities. However, if the computation time of a task is larger than a threshold, its priority will become lower. This is because a task with a large computation time may block other tasks for a long time.

- Step 3: signal packing. The signal packing is done in a greedy fashion. For each signal, we try to find a message satisfying the following conditions: the signal and the message have the same period, the source task of the signal is allocated to the source ECU of the message, and there is enough payload space for the signal and its MAC (we assume that each receiving ECU has its own receiving group). If we find a message satisfying the conditions, the signal is packed into the message; otherwise, the signal is packed into an empty message.

- Step 4: message priority assignment. The message priority is also assigned according to the Rate-Monotonic Policy, *i.e.*, messages with smaller periods have higher priorities.

- Step 5: receiving group assignment. The receiving group assignment is done in a greedy fashion. For each receiving ECU of a message, we try to find a receiving group of the message such that adding the ECU into the receiving group does not violate the security constraint for direct attacks, as formulated in Equation (4.47). If we find a receiving group satisfying the condition, the ECU is assigned to the receiving group, and the MAC length of the receiving group may need to be increased to satisfy the security constraint for indirect attacks, as formulated in Equation (4.48); otherwise, the ECU is assigned to an empty receiving group. Note that we assume that each receiving ECU has its own receiving group in Step 3, so assigning two receiving ECUs to the same receiving group never increases the length of the message.

4.4 Experimental Results

We obtained an industrial test case [56]. The test case supports advanced distributed functions with end-to-end computations collecting data from 360-degree sensors to the actuators, consisting of the throttle, brake and steering subsystems and of advanced Human-Machine Interface devices. The architecture platform consists of 9 ECUs connected through a single CAN [3] or CAN-FD [4] bus with the speed 500 *kbps*. The functional model consists of 41 tasks and 83 signals. For the safety requirements, 171 paths are selected with deadlines 300 *ms* or 100 *ms*. For the security requirements, 50 signals are selected with required MAC lengths ranging from 10 to 30 bits for CAN and from 64 to 128 bits for CAN-FD (with longer message length, CAN-FD is able to provide more MAC bits and therefore more secure communications). The maximum allowed security risk of each signal is simplified so that no

Table 4.4: The objective (the summation of latencies of selected paths), maximum latencies, load, and runtime of each step of the MILP-based algorithm, where "Max L_{300}" and "Max L_{100}" are the largest latencies among the paths with deadlines 300 *ms* and 100 *ms*, respectively.

	Step (X)	Results after Step X				
		Objective (*ms*)	Max L_{300} (*ms*)	Max L_{100} (*ms*)	Bus Load (*kbps*)	Runtime (*s*)
CAN	1	11,070.61	127.92	90.72	76.92	3,600
	2	11,069.88	127.82	90.62	45.57	< 600
	3	11,069.62	127.79	90.59	31.52	< 10
CAN-FD	1	11,075.08	128.56	91.22	211.74	3,600
	2	11,073.67	128.39	91.05	176.47	< 600
	3	11,071.69	128.14	90.80	98.33	< 10

more than 2 ECUs can be assigned to the same receiving group, *i.e.*, $2 \leq \frac{R_{i,j}}{R_{i,j,k',k''}} < 3$ in Equation (4.16). The program is implemented in C/C++. CPLEX 12.5 is used as the MILP solver. The experiments are run on a 2.5-GHz processor with 4GB RAM. We compare our MILP-based algorithm with a greedy heuristic and non-integrated approaches applying the pair-wise key distribution and the one-key-for-all key distribution.

4.4.1 Comparison with a Greedy Heuristic

For comparison with the MILP-based algorithm, we also implement a greedy heuristic. First, we calculate a weight between each task pair that represents an estimation of how much benefit we can gain by making the signals between the two tasks become local signals (*i.e.*, mapping the two tasks onto the same ECU). This estimation depends on the potential gain from reducing the path latency and the potential gain on security (local signals are not at risk of masquerade and replay attacks). Then, we try to cluster the tasks and assign them onto the same ECU. We check whether we can assign two tasks onto the same ECU without violating utilization constraints (if the two tasks are already assigned to different ECUs, we check whether we can assign all the tasks from these two ECUs onto one ECU). Once task allocation is decided, we pack the signals into messages in a greedy fashion, *i.e.*, we merge two messages as long as the combined message satisfies size constraints and we merge MACs as long as the security constraints permit. Finally, we assign priorities based on the Rate Monotonic policy.

The results are listed in Table 4.4. For the CAN protocol, our MILP-based algorithm can find a solution satisfying all the design constraints. In Step 1, within 3,600 seconds, the objective of the best solution found by the solver is 11,070.61 *ms*. The largest latencies among the paths with deadlines 300 *ms* and 100 *ms* are 127.92 *ms* and 90.72 *ms*, respectively. In Step 2, the program ends in 600 seconds, and the objective is 11,069.88 *ms*. The largest latencies among the paths with deadlines 300 *ms* and 100 *ms* are 127.82 *ms* and 90.62 *ms*,

respectively. In Step 3, the program ends in few seconds, and the objective is 11,069.62 *ms*. The largest latencies among the paths with deadlines 300 *ms* and 100 *ms* are 127.79 *ms* and 90.59 *ms*, respectively. There is little improvement on the latency objective in Steps 2 and 3 because the message response times are much smaller compared to the task and message periods that contribute to the path latencies in our model. However, Steps 2 and 3 can significantly reduce the bus load from 76.92 *kbps* to 45.57 *kbps* and 31.52 *kbps*, respectively.

In comparison, the objective of the greedy heuristic is 23,114.50 *ms*, while satisfying all the design constraints. Its runtime is 1.4 seconds, but the value of the objective is much worse than the one obtained with the MILP-based algorithm. This is because the exploration is stuck at a local minimum, which is a common problem for heuristic algorithms.

For the CAN-FD protocol, the MILP-based algorithm can also find a solution satisfying all the design constraints. The objectives, the largest latencies, and the bus loads are increased because the required MAC lengths are much longer (128 bits to 64 bits). Similarly, Steps 2 and 3 reduce the bus load from 211.74 *kbps* to 176.47 *kbps* and 98.33 *kbps*, respectively, showing the effectiveness of signal packing and our flexible key distribution scheme. On the other hand, the greedy heuristic cannot find a feasible solution in this case (with bus speed at 500 *kbps*) due to the much longer required MAC lengths. In fact, we find that the heuristic can only find a feasible solution when we increase the bus speed to 4,000 *kbps*.

4.4.2 Comparison with Non-Integrated Approaches

We also tried two experiments in which we do not consider security constraints explicitly in Steps 1 and 2 (*i.e.*, solving a traditional mapping problem with only timing constraints), and then explore the addition of MAC bits and the key distribution in Step 3. In the first experiment, in Steps 1 and 2, we constrain that all messages should have at most 32 bits used for data while packing signals, *i.e.*, leaving 32 bits available for MAC bits. Then, in Step 3, we find that there is no feasible solution for either the pair-wise key distribution or the one-key-for-all key distribution. The reason is that the pair-wise key distribution requires more than 32 MAC bits for certain messages, while the one-key-for-all key distribution leads to too high security risks for some messages. In the second experiment, in Steps 1 and 2, we do not set any constraint on the number of bits for data, *i.e.*, they may use as many as all 64 bits in the payload. Then, in Step 3, we find that there is no feasible solution for the pair-wise key distribution, the one-key-for-all key distribution, or our flexible key distribution scheme. This is because some messages use almost all 64 bits, so no MAC can be added to those message. The results from these two experiments demonstrate that it is necessary to consider security together with other metrics during mapping; otherwise, it may be difficult or even impossible to add security measurement later.

4.4.3 Extension

There are three settings for both of CAN and CAN-FD. The first one is the same as that above. For the security requirements, 50 signals are selected with required MAC lengths

Table 4.5: The comparison between the MILP-based approach, the greedy heuristic, and the extended algorithm. "×" means "no feasible solution." In the third setting, an objective includes the risks of direct attacks and indirect attacks.

			Security Constraints & Objective Function		
			Signal-Based & Latency Min.	Path-Based & Latency Min.	Path-Based & Security Risk Min.
CAN	MILP-Based Approach	Objective (ms)	11,069.62	13,420.64	0 & 7.05×10^{-3}
		Runtime (s)	$> 3{,}600$	$> 3{,}600$	$> 3{,}600$
	Greedy Heuristic	Objective (ms)	23,114.50	—	—
		Runtime (s)	< 1.5	—	—
	Extended Algorithm	Objective (ms)	13,789.64	13,789.47	0 & 6.05×10^{-3}
		Runtime (s)	< 0.5	< 0.5	< 0.5
CAN-FD	MILP-Based Approach	Objective (ms)	11,071.69	13,422.59	0 & 3.79×10^{-19}
		Runtime (s)	$> 3{,}600$	$> 3{,}600$	$> 3{,}600$
	Greedy Heuristic	Objective(ms)	×	—	—
		Runtime (s)	×	—	—
	Extended Algorithm	Objective (ms)	13,863.31	13,683.14	0 & 3.25×10^{-19}
		Runtime (s)	< 0.5	< 0.5	< 0.5

ranging from 10 to 30 bits for CAN and from 64 to 128 bits for CAN-FD. The maximum allowed security risk of a signal is that no more than 2 ECUs can be assigned to the same receiving group. The second setting replaces the signal-based security constraints by the path-based security constraints, where 30 paths are selected, and their required MAC lengths ranging from 10 to 30 bits for CAN and from 64 to 128 bits for CAN-FD. The maximum allowed security risk of a path is that, considering all signals in the path, no more than 2 to 5 ECUs can be assigned to the same receiving group. The third setting replaces the objective function in the second setting by minimizing security risk as Equation (4.49).

Table 4.5 shows the comparison between the MILP-based approach, the greedy heuristic (it cannot deal with path-based security constraints), and the extended algorithm. In the first setting, the extended algorithm gets the objective 13,789.64 ms, which is larger than that of the MILP-based approach. However, the runtime is less than 0.5 second, while the MILP-based approach (which cannot find a feasible solution in such a short time) takes more than 1 hour. Note that the MILP-based approach terminates by timeouts, so it needs more time for a complete run. On the other hand, the extended algorithm significantly outperforms the greedy heuristic.

In the second setting, the extended algorithm gets the comparable objectives as the MILP-based approach and takes much less runtime. The solution quality is almost the same because the extended algorithm with the connectivity concept finds a good task allocation efficiently. On the other hand, although the MILP formulation is optimal for the sub-problem of task allocation and task priority, its runtime is too long. Therefore, when it is interrupted by timeouts, its solution quality is not significantly better than that of the extended algorithm. This indicates that task allocation and task priority assignment are

very crucial for the objective minimization, and adding a good searching algorithm to the extended algorithm may lead to even better results (part of our future work).

In the third setting, an objective includes the risks of direct attacks and indirect attacks. The risk of direct attacks is defined in Equation (4.49), and the risk of indirect attacks is the summation of the probability that an attacker guesses a MAC correctly. The risks of direct attacks are minimized to 0 because each receiving ECU has its own secret keys and MACs, and the higher overheads do not violate any constraint. The extended algorithm has lower risks of indirect attacks because its task allocation tends to allocate the source and destination tasks of a signal in a security-critical path to the same ECU. Given that there is no direct attack, we can increase the MAC lengths to reduce the risk of indirect attacks. In this case, we can further reduce the risks of indirect attacks from 6.05×10^{-3} and 3.25×10^{-19} to 1.53×10^{-5} and 1.09×10^{-37} (almost every MAC can be up to 128 bits with CAN-FD), respectively.

4.5 Summary

In this chapter, we propose an approach to address both the security and the safety in the design space exploration of CAN-based systems. We present an MILP formulation that explores task allocation, signal packing, MAC sharing, and priority assignment while meeting both security and safety constraints. We further present an extended formulation that defines path-based security constraints and minimizes security risk directly and develop a new heuristic algorithm to solve the problem more efficiently. Experimental results of an industrial case study show that our approaches can effectively and efficiently explore the design space to meet the system security and safety requirements.

Chapter 5

Security-Aware Mapping for TDMA-Based Systems

In this chapter, we focus on security-aware design for Time Division Multiple Access (TDMA) based real-time distributed systems. The TDMA-based protocol is a very representative synchronous protocol and an abstraction of many existing protocols, such as the FlexRay [7], the Time-Triggered Protocol [42], and the Time-Triggered Ethernet [41]. It is critically important to address these protocols, as they are being increasingly adopted in various safety-critical systems such as automotive and avionics electronic systems for their more predictable timing behavior. Compared with priority-based networks such as the Controller Area Network (CAN) protocol, TDMA-based systems have fundamental differences on system modeling (in particular for latency modeling), on security mechanism selection (a global time is available for security reasons), on design space (network scheduling is the focus of this work but not a factor for CAN-based systems), and on algorithm design. Therefore, the approaches for CAN-based systems in the previous chapter do not apply to TDMA-based systems. We need to rethink appropriate security mechanisms and develop a new set of formulations and algorithms to explore the design space.

There are many security mechanisms that can be applied to the TDMA-based protocol. For message authentication, legitimate senders and receivers usually share keys so that they can use the keys to compute Message Authentication Codes (MACs) and protect against masquerade attacks. As mentioned in Chapter 2, key management strategies are divided into several categories [52]: one-key-for-all key distribution, pair-wise key distribution, asymmetric cryptography, and time-delayed release of keys [2]. The one-key-for-all key distribution is simple but not suitable for distributed systems because it does not protect against masquerade attacks from a node in the group. The pair-wise key distribution protects against such masquerade attacks, but it has limited scalability because the message size increases quickly with the number of nodes in the network. The approach of asymmetric cryptography (private and public keys) provides higher security level, but its computational overhead is much higher with the usage of asymmetric ciphers, which makes it difficult to be used in resource-limited real-time distributed systems. Compared with these three approaches, the

approach of time-delayed release of keys is the most suitable for real-time distributed embedded systems because it provides a good balance between security level and computation and communication overhead [52]. As an example, the Timed Efficient Stream Loss-Tolerant Authentication (TESLA) [34, 35] security mechanism, is based on the time-delayed release of keys.

Despite being a more economical choice, using the time-delayed release of keys for TDMA-based systems still puts significant timing overheads on communication and computation in real-time embedded systems. In particular, the message latencies may significantly increase due to the waiting for key releases, and the end-to-end latencies may violate deadline requirements. For system safety and performance, it is critical to ensure that the usage of such security mechanism will not violate any timing constraint.

We apply message authentication with the time-delayed release of keys to protect against attacks on a TDMA-based protocol, and develop formulations and an algorithm to explore the design space while meeting both the security and the timing requirements. Specifically for the exploration, we optimize the task allocation, priority assignment, network scheduling, and key-release interval length during the mapping process from the functional model to the architectural model, while considering the overhead of the security mechanism and end-to-end deadline constraints. To the best of our knowledge, this is the first work to address both the security and safety constraints during the system level mapping process for the time-delayed release of keys for TDMA-based real-time distributed systems.

We develop an algorithm that combines simulated annealing with a set of efficient optimization heuristics for security-aware mapping. In particular, we propose a network scheduler and a transmission delay analyzer (which outputs exact solutions in a single-switch network) to optimize the network scheduling and analyze the worst-case transmission delay. Our network scheduler and latency analyzer can address synchronous and asynchronous message arrivals, both of which are common scenarios in real-time distributed systems, *e.g.*, they match the Time-Triggered (TT) messages and the Rate-Constrained (RC) messages in the Time-Triggered Ethernet protocol. Experimental results of an industrial case study and a synthetic example show the effectiveness and efficiency of our algorithm, and demonstrate that security must be considered with other metrics at the design stage.

The chapter is organized as follows. Section 5.1 defines the system model. Section 5.3 presents the mapping algorithm. Section 5.4 demonstrates the experimental results. Section 5.5 provides a summary of this chapter.

5.1 System Model

The mapping problem addressed in this chapter is also based on Platform-Based Design paradigm [44] and the methodology in Chapter 2, where the functional model and the architecture platform are initially captured separately and then brought together through a mapping process, meaning that the functional model is implemented on the architecture plat-

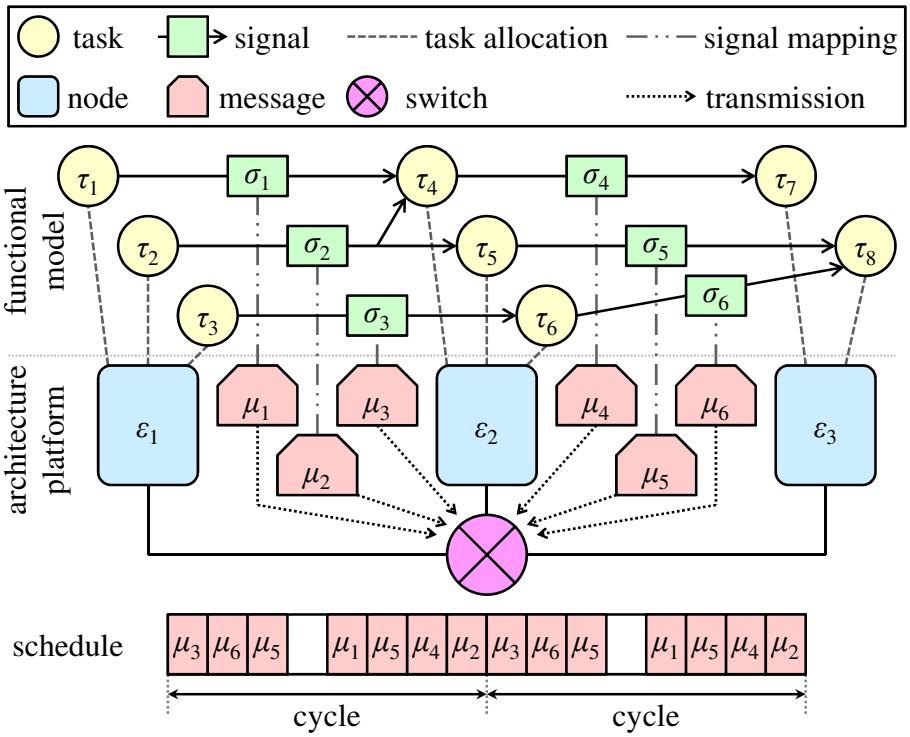


Figure 5.1: The system model of a TDMA-based system.

form. Besides, there are objectives and constraints in the mapping process to be optimized and satisfied.

As shown in Figure 5.1, similar to the system model in Chapter 4, the functional model is a task graph that consists of a set of tasks, denoted by $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_{|\mathcal{T}|}\}$, and a set of signals, denoted by $\mathcal{S} = \{\sigma_1, \sigma_2, \ldots, \sigma_{|\mathcal{S}|}\}$. Each signal is between a source task and a destination task. Tasks are activated periodically and communicate with each other through signals. The architecture model is a distributed platform that consists of a set of computation nodes, denoted by $\mathcal{E} = \{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{|\mathcal{E}|}\}$, and nodes are assumed to support preemptive priority-based task scheduling. The nodes are connected through a TDMA-based switch (we focus on the single-switch case in this chapter, and our formulation can be extended to multi-switches cases). A set of messages is communicated among nodes through the switch, denoted by $\mathcal{M} = \{\mu_1, \mu_2, \ldots, \mu_{|\mathcal{M}|}\}$. The switch uses a TDMA-based model for scheduling, in which each *time slot* in the schedule can be assigned to one message. Several time slots form a *cycle*, and the network switch repeats the same scheduling sequence after each cycle. It is possible that a time slot is empty (not assigned to any message) in a schedule, and it is also possible that there are more than one time slots assigned to the same message in a cycle.

During mapping, the functional model is mapped onto the architecture platform, as shown in Figure 5.1. Specifically, the tasks are allocated onto nodes with their priorities,

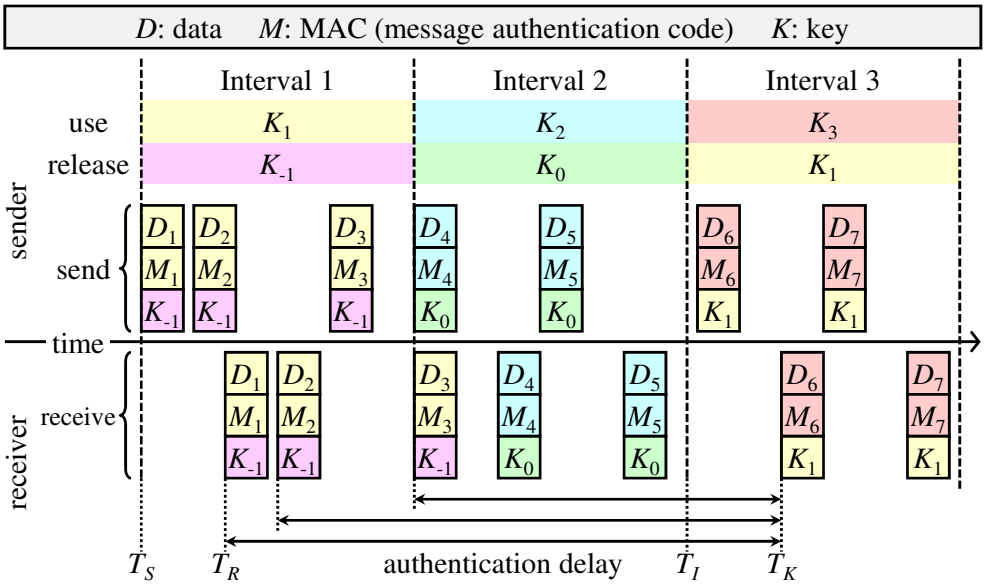


Figure 5.2: The time-delayed release of keys. T_S, T_R, and T_K are the sending time, the receiving time, and the key-receiving time of the packet (D_1, M_1, K_{-1}), respectively. T_I is the starting time of Interval 3.

and the signals are mapped onto messages and transmitted on the network. Messages are triggered periodically and each message contains the latest values of the signals that are mapped to the message.

We consider three possible types of attacks [52], including tapping the port of an existing node, replacing an existing node, and connecting to an empty port of the switch. The time-delayed release of keys [2, 34, 35, 52] as introduced in the next section is used to prevent these attacks.

5.2 Time-Delayed Release of Keys

Definition 5.1. A *packet* is an instance of a message.

The time-delayed release of keys is a popular approach for message authentication. In this security mechanism, each sender maintains a key chain where the keys in the key chain are computed in a reversed order to provide fault tolerance. Usually, keys in the key chain are not used for computing MACs. Instead, they are used for computing other keys, and those keys are used for computing MACs [34, 35]. A sender maintains *intervals*, and it uses the same key to compute MACs in one interval. When the sender intends to send a packet in an interval, it uses the corresponding key in the interval to compute a MAC and sends the packet including the data, the MAC, and the key used in several intervals before. When a receiver receives the packet, it stores the data and the MAC first. Once the receiver receives

the corresponding key (which will be released by the sender after several intervals), it can authenticate the packet.

An example of the time-delayed release of keys is shown in Figure 5.2, where we show the keys used for computing MACs and the released keys. When the sender intends to send data D_1 in Interval 1, it uses key K_1 to compute MAC M_1 and sends (D_1, M_1, K_{-1}), where K_{-1} is the key used in two intervals before. The receiver receives it and stores D_1 and M_1 first. In Interval 3, the sender sends (D_6, M_6, K_1). When the receiver receives the packet, it uses K_1 to authenticate (D_1, M_1, K_{-1}).

Definition 5.2. Given a packet P, the *sending time* T_S of the packet is the time that its sender sends it. The *receiving time* T_R of the packet is the time that its receiver receives it. The *key-receiving time* T_K of the packet is the time that its receiver receives its corresponding key (for the first time).

The *security requirement of the time-delayed release of keys* [34, 35]: a packet is safe if its receiving time is before the moment its corresponding key may be released (otherwise masquerade attacks can be conducted), *i.e.*, for each packet,

$$T_R < T_I, \tag{5.1}$$

where T_I is the starting time of the interval in which the corresponding key for the packet is released[1].

In the example in Figure 5.2, since the sender uses K_1 to compute M_3 for the packet (D_3, M_3, K_{-1}) and the packet arrives at the receiver in Interval 2, the sender has to wait until Interval 3 to release K_1.

Definition 5.3. Given a packet P, the *transmission delay* D_T of the packet is its receiving time minus its sending time, *i.e.*, $D_T = T_R - T_S$. The *authentication delay* D_A of the packet is its key-receiving time minus its receiving time, *i.e.*, $D_A = T_K - T_R$. The *latency* L of the packet is its key-receiving time minus its sending time, *i.e.*, $L = T_K - T_S = D_T + D_A$.

Compared with traditional symmetric ciphers, the time-delayed release of keys has a lower computational overhead because a sender only needs to compute one MAC for each packet. This is because, if there are n receivers, a symmetric cipher needs n MACs, while the time-delayed release of keys only needs 1 MAC. Compared with asymmetric ciphers which have more complex calculation, the time-delayed release of keys also has a much lower computational overhead [52][2]. However, as shown in Figure 5.2, it increases the latency of a packet due to the authentication delay. In Section 5.3.4, we will show how the network scheduling plays an important role in reducing the latency of a packet, which is extremely critical for real-time distributed systems.

[1] If the synchronization precision is considered, we can add a small positive constant (the precision of time) to the left-hand side.

[2] The most common comparison is between a Keyed-Hash Message Authentication Code (HMAC) and the RSA algorithm, and HMAC (based on XOR and hash function) is usually much faster than RSA (based on exponentiation and modular arithmetic).

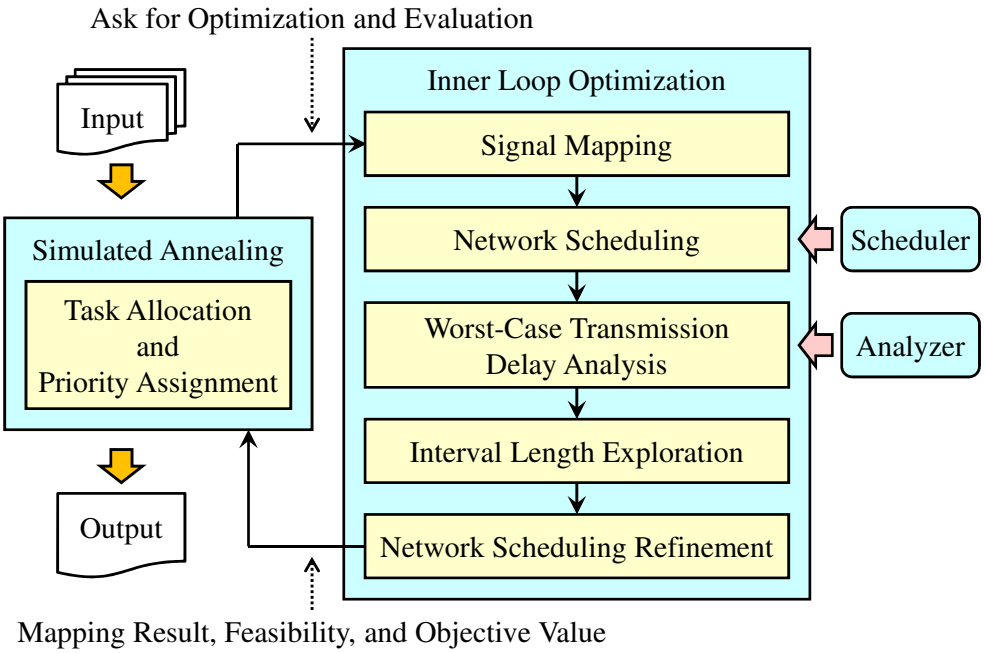


Figure 5.3: The algorithm flow.

5.3 Mapping Algorithm

Given a system, we explore the design space of task allocation, priority assignment, signal mapping, network scheduling, and interval length. The end-to-end deadline requirements are set on a set of time-critical functional paths. The worst-case latency of a time-critical path should not be larger than its deadline. The optimization objective is to minimize the summation of the worst-case latencies of all time-critical paths.

5.3.1 Overview

Figure 5.3 shows the flow of our algorithm. It combines simulated annealing with a set of optimization heuristics. In the simulated annealing, the task allocation and the task priority are randomly changed. Every time the task allocation and the task priority are changed, the algorithm calls the inner loop optimization to perform a set of optimization heuristics and evaluate the feasibility and the objective value. The inner loop optimization consists of five steps: signal mapping, network scheduling, worst-case transmission delay analysis, interval length exploration, and network scheduling refinement. After these five steps, the inner loop optimization returns the mapping result (signal mapping, network schedule, and interval length). It also returns the feasibility and the objective value, which are used by the simulated annealing to decide whether to keep the changed task allocation and task priority or not. We decide the task allocation and the task priority first in the simulated annealing

because they have significant impact on the design constraints and objectives, and also on the possible values for other design variables. The inner loop optimization is called every time the task allocation and the task priority are changed, so it must be very efficient and effective. The details are introduced in the following sections.

Note that we mostly target on design-time security-aware mapping optimization for a given task set and a given hardware platform. The inner loop optimization in the algorithm is very efficient, and can be applied at runtime if there are dynamic changes on task set or hardware platform. The simulated annealing part is not suitable for runtime optimization because of its long running time. Exploring priority assignment at runtime will need efficient heuristics, and task (re)-allocation at runtime may be infeasible in practice because of its overhead.

5.3.2 Task Allocation and Priority Assignment

The initial allocation of a task is assigned based on the task index modulo the number of nodes, *i.e.*, tasks are distributed as evenly as possible. The initial priority of a task is assigned in a greedy fashion—the tasks that appear in more time-critical paths are assigned with higher priorities. During simulated annealing, two random operations may be performed. The first one is to allocate a task to another node, and the second one is to switch the priorities of two tasks. We use a parameter P to control the probability that the first operation is selected in each iteration, while the probability that the second operation is selected is $1 - P$.

To explore the design space more efficiently, we also propose an accelerating method for the simulated annealing. With this accelerating method, tasks are divided into two groups, depending on whether tasks are in time-critical paths or not. Tasks in the first group are in at least one time-critical path, and tasks in the second group are not in any time-critical path. If the first operation is selected, there is another parameter Q to control the probability that a task in the first group is selected. This method can effectively accelerate the simulated annealing because those tasks in the first group play more important roles in the constraint satisfaction and the objective minimization.

5.3.3 Signal Mapping

Each signal needs to be mapped onto a message, and we assume each signal is packed into its own message in this chapter. Without loss of generality, we assume that signal σ_j is mapped to message μ_j, so the period, the length, the source node, and the destination node of a message can be directly decided. What we need to explore is whether a message should be *synchronous* or *asynchronous*.

Definition 5.4. For a *synchronous* message, the network knows the time that each packet of the message is sent.

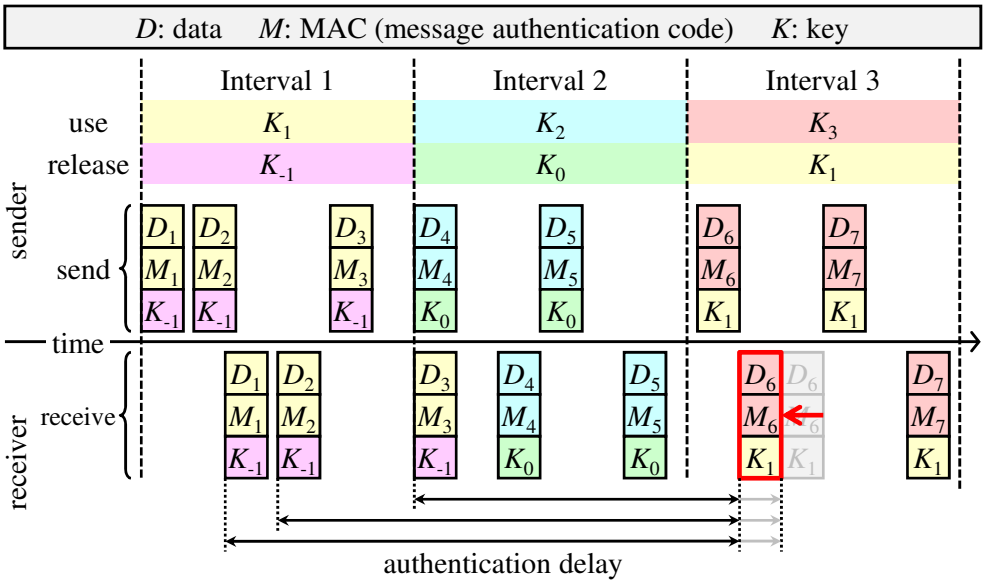


Figure 5.4: An approach to reduce the authentication delay.

Definition 5.5. For an *asynchronous* message, the network does not know the time that each packet of the message is sent but knows the period (or pattern) of the message.

In our algorithm, if a signal is time-critical (the signal is on at least one time-critical path), then its message is assigned as a synchronous message; otherwise, its message is assigned as an asynchronous message. If a message is synchronous, we also need to decide the time that the first packet of the message is sent. For message μ_j, we assign the time that the first packet of the message is sent as $j \times L$ modulo T_j^σ, where L is the time length of a time slot and T_j^σ is the period of σ_j. This assignment can lower the probability that the packets of two different messages are sent at the same time. If this happens, one packet will be delayed, and the transmission delay of its message may increase. It should be mentioned that, in Time-Triggered Ethernet [41], a synchronous message matches the Time-Triggered (TT) traffic, while an asynchronous message matches the Rate-Constrained (RC) traffic. If a network only supports synchronous messages or asynchronous messages, then all messages are assigned as synchronous messages or asynchronous messages.

5.3.4 Network Scheduling

To satisfy design constraints, the increased latency due to the authentication delays must be considered and reduced at the design stage. We observe that there are three approaches for the network scheduling that can reduce packet latency.

The first approach is that a scheduler may schedule each sender's first packet within an interval so that it can be received earlier. The key-receiving time of a packet P is the

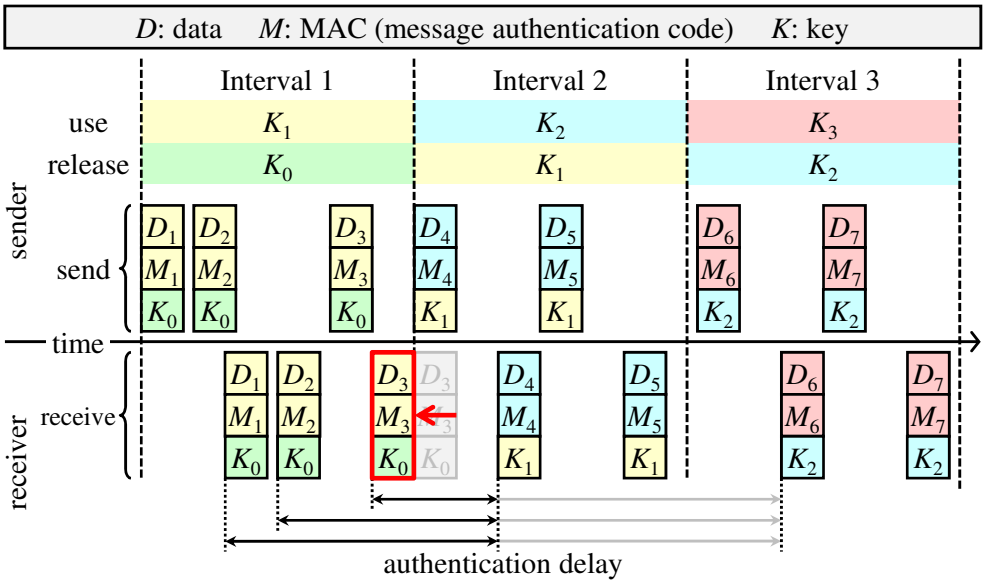


Figure 5.5: A more effective approach to reduce the authentication delay.

receiving time of the first packet P' carrying the corresponding key, so the latency of P is

$$L = T_K - T_S = T'_R - T_S = T'_S + D'_T - T_S, \tag{5.2}$$

where T'_S, T'_R, and D'_T are the sending time, the receiving time, and the transmission delay of packet P'. The first approach minimizes L by minimizing D'_T. As shown in Figure 5.4, M_1, M_2, and M_3 are computed by K_1. Because the receiver receives the packet (D_6, M_6, K_1) earlier, it can authenticate the packets (D_1, M_1, K_{-1}), (D_2, M_2, K_{-1}), and (D_3, M_3, K_{-1}) earlier, and their authentication delays and latencies become smaller, compared with the timing illustrated in Figure 5.2.

The second approach is that a scheduler may try to schedule a packet earlier to ensure that it is received before the end of the interval. As a result, the sender can release keys one interval earlier without violating the security requirement, and the authentication delays and latencies become smaller. The first packet P' carrying the corresponding key is sent after the starting time of the corresponding interval, so

$$T_I < T'_S. \tag{5.3}$$

Note that T_I is the starting time of the interval in which the corresponding key for the packet is released. The second approach minimizes T_I first so that T'_S can also be minimized (the corresponding key can be released earlier). From Equation (5.2), the latency of P becomes smaller. As shown in Figure 5.5, because the receiver receivers the packet (D_3, M_3, K_0) before the end of Interval 1, the sender can release keys just one interval earlier without violating the security requirement. As a result, the authentication delays and latencies become smaller, compared with the timing illustrated in Figure 5.2.

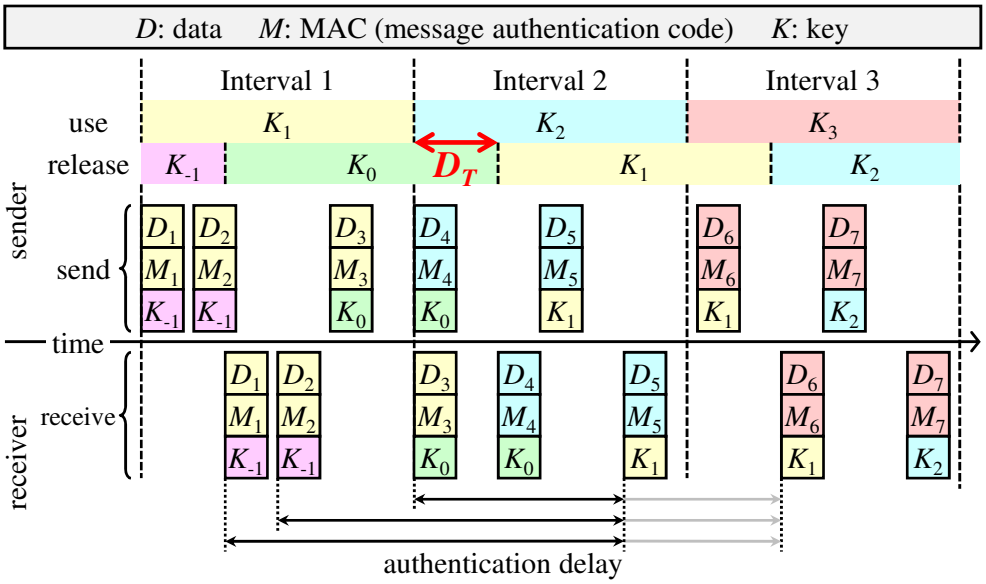


Figure 5.6: Given the worst-case transmission delay D_T, an approach to reduce the authentication delay, where the key-using and key-releasing intervals are not aligned.

The third approach is that a scheduler may minimize the worst-case transmission delay of packets. In some cases, if a scheduler cannot schedule a packet so that it is sent and received in the same interval (for example, a packet is sent just before the end of an interval), the second approach cannot work. However, it provides an insight that, if a scheduler can minimize the worst-case transmission delay of packets, keys can be released earlier. In the third approach, different from the traditional design of the time-delayed release of keys, the intervals of used keys and released keys are not aligned. As shown in Figure 5.6, given the worst-case transmission delay D_T, the key-using and key-releasing intervals are not aligned, and a key is released D_T time units after the end of the interval in which the key is used for computing MACs. As a result, the authentication delay and the latency are also reduced, compared with the timing illustrated in Figure 5.2. Combining Definition 5.2 and Equations (5.1) and (5.3), we get

$$T_R = T_S + D_T < T_I < T'_S. \tag{5.4}$$

The third approach minimizes D_T first so that T_I and then T'_S can also be minimized. From Equation (5.2), the latency of P becomes smaller.

We will reduce the latency of a packet through the above three approaches. In this step, we minimize the worst-case transmission delay of packets so that keys can be released earlier (the second and the third approaches[3]). We will then try to release keys earlier in the network scheduling refinement step in Section 5.3.7 (the first approach).

[3]Note that the second approach can be regarded as a special case of the third approach. Both of them try to minimize transmission delays of packets.

Specifically, in this step, we first assign priorities to messages. A message whose corresponding signal appears more times in the time-critical paths is given a higher priority. We then schedule messages one-by-one according to their priorities. If message μ_j is synchronous, we schedule time slots to it "as early as possible". In other words, we schedule the first time slot after the arrival of a packet to the message. If message μ_j is asynchronous, we first compute the number of time slots that we plan to allocate to the message in a cycle. For asynchronous message μ_j, the number of time slots in a cycle is $\left\lceil R \times \frac{N \times L}{T_j^\sigma} \right\rceil$, where R is a parameter larger than or equal to 1, N is the number of time slots in a cycle, L is the time length of a time slot, and T_j^σ is the period of σ_j. After computing the number of time slots, we schedule time slots to the asynchronous message "as evenly as possible". It is possible that a time slot has been used (occupied) by a higher-priority message. In this case, we schedule the next empty time slot to the message. It is very important that we schedule synchronous messages as early as possible and asynchronous messages as evenly as possible, as they are the optimal strategies for each of them (which will be further demonstrated in the next section).

One thing that should be emphasized is the choice of R value. A large R means that denser time slots are scheduled to an asynchronous message, and the worst-case transmission delay of the message may decrease. If we do not consider the time-delayed release of keys, the decreasing of the worst-case transmission delay of an asynchronous message has no effect on the objective value because only non-time-critical signals are mapped to an asynchronous message. This is also a reason that a traditional scheduler may not be suitable for this problem. On the contrary, when we consider the time-delayed release of keys in this case, the decreasing of the worst-case transmission delay of an asynchronous message enables its sender to release keys earlier, so the worst-case latencies of synchronous messages and the objective value can become smaller. Therefore, we increase the parameter R in our case. Specifically, if $R = 1$ and the network utilization rate (the ratio of the number of scheduled time slots to the number of total time slots) is smaller than a pre-assigned value U, we increase R so that the network utilization rate reaches U.

5.3.5 Worst-Case Transmission Delay Analysis

Besides network scheduling, an accurate analyzer for computing the worst-case transmission delay is also very important. Given the worst-case transmission delay D_T, a key can be released D_T time units after the end of the interval in which the key is used for computing MACs. If the analyzer underestimates the worst-case transmission delay, the security requirement may be violated because keys may be released too early. If the analyzer overestimates the worst-case transmission delay (*i.e.*, being too pessimistic), minimizing the worst-case transmission delay may not be effective. To compute the worst-case transmission delay, we first define the packet arrival pattern and the schedule pattern of a message as follows:

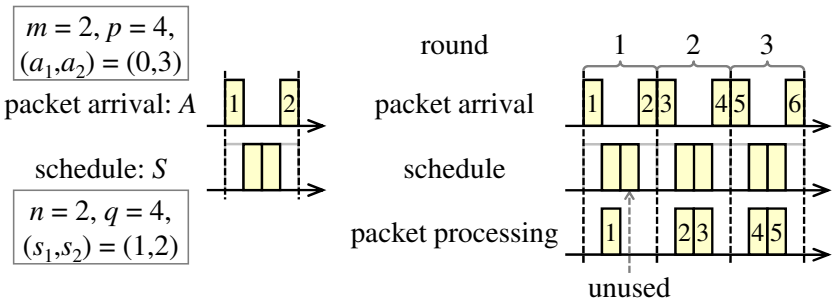


Figure 5.7: An example for synchronous messages with tree rounds. The second packet (#2, #4, or #6) of each round is an unscheduled packet after its corresponding round, and the second time slot of the first round is an unused time slot. The second round and the third round have the same packet processing pattern.

Definition 5.6. A *packet arrival pattern* A is defined by m, p and $a_1, a_2, \ldots, a_m$, where the arriving times of packets are $a_1, a_2, a_3, \ldots, a_m$, and the pattern repeats with a period p ($a_i < p$ for all i, $1 \leq i \leq m$).

Definition 5.7. A *schedule pattern* S is defined by n, q and $s_1, s_2, \ldots, s_n$, where the starting times of time slots are $s_1, s_2, s_3, \ldots, s_n$, and the pattern repeats with a period q ($s_i < q$ for all i, $1 \leq i \leq n$).

Assumption 5.1. One time slot serves one packet.

The problem here is: "given the packet arrival pattern A and the schedule pattern S of a message, what is the worst-case transmission delay of the packet arrival pattern?" We will discuss synchronous messages and asynchronous messages in the following sections.

5.3.5.1 Synchronous Message

Definition 5.8. A *round* is a time period whose length is the least common multiple of p and q. A packet is *unscheduled* after a round if it is not assigned to any time slot after the round; otherwise, it is *scheduled*. A time slot is *unused* if no packet is assigned to it; otherwise, it is *used*.

In Figure 5.7, there are three rounds. The second packet of each round is an unscheduled packet after its corresponding round. The second time slot of the first round is an unused time slot. Given a synchronous packet arrival pattern A and its schedule pattern S, we only need to consider two rounds for the worst-case transmission delay of the packet arrival pattern. In the analysis, we start from the first packet and assign each packet to the first unused time slot after the arrival of the packet.

Theorem 5.1. *We only need to consider two rounds for the worst-case transmission delay of the packet arrival pattern.*

Proof. We claim that the numbers of unscheduled packets after the first round and the second round are the same. Therefore, the pattern of the second round is the same as that of any following round. We will prove that the number of unscheduled packets does not decrease or increase after the second round. For the non-decreasing part, it is because the second round is more difficult (with some unscheduled packets after the first round) than the first round. For the non-increasing part, we first assume that $\frac{m}{p} \leq \frac{n}{q}$, *i.e.*, the number of packets in a round is never larger than the number of time slots in a round; otherwise, the algorithm returns infinity directly. Given this assumption, after the first round, the number of unscheduled packets is never larger than the number of unused time slots. Accordingly, in the second round, the repeated time slots of those unused time slots in the first round are sufficient for those unscheduled packets in the first round. Besides, the repeated time slots of those used time slots in the first round are still sufficient for the repeated packets of those scheduled packets in the first round. Therefore, the number of unscheduled packets does not increase. Combining the two parts, the numbers of unscheduled packets after the first round and the second round are the same, so we only need to consider the first two rounds. □

The theorem also implies that an unscheduled packet after the second round does not affect the result. An example is shown in Figure 5.7, where the second round and the third round have the same packet processing pattern. It also shows that we need to consider at least two rounds for the worst-case transmission delay of the packet arrival pattern.

5.3.5.2 Asynchronous Message

Definition 5.9. A packet *just misses* a time slot if the starting time of the time slot is ϵ time unit earlier than the arriving time of the packet where $\epsilon \to 0$.

Theorem 5.2. *If the worst-case transmission delay happens when packet P_i is assigned to time slot S_j, then (1) one of P_i itself and the packets arriving before P_i must have just missed a time slot, and (2) there must be no unused time slot between the arriving time of the packet just missing a time slot and the starting time of S_j.*

Proof. For the first part, if all of P_i and the packets arriving before P_i do not just miss a time slot, then we can shift all packets so that they arrive earlier but are assigned to the same time slots. As a result, the transmission delays of them become larger, which is a contradiction. For the second part, there are two cases. The first case is that P_i itself just misses a time slot. If there is an unused time slot between the arriving time of P_i and the starting time of the S_j, then P_i should be assigned to the unused time slot, which is a contradiction. The second case is that one of the packets arriving before P_i just misses a time slot. If there is an unused time slot between the arriving time of the packet just missing a time slot and the starting time of S_j, then we can define the packet which is assigned to the first used time slot after the unused time slot as P_k. Next, we can shift all packets so that they arrive earlier but all packets arriving between P_k and P_i are assigned to the same time slots. As a result, the transmission delays of them become larger, which is a contradiction. □

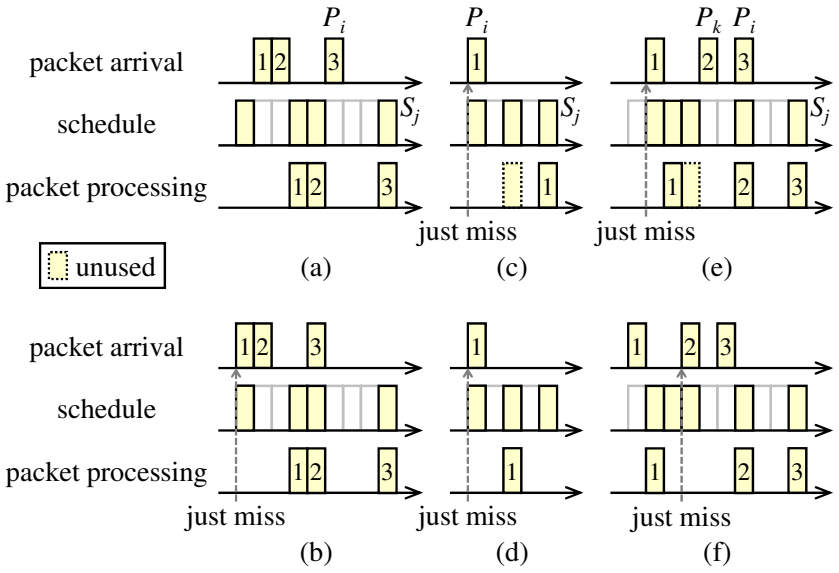


Figure 5.8: For asynchronous messages, if the worst-case transmission delay happens when packet P_i is assigned to time slot S_j, then (a–b) one of P_i itself and the packets arriving before P_i must just miss a time slot, and (c–f) there must be no unused time slot between the arriving time of the packet just missing a time slot and the starting time of S_j.

Figure 5.8(a) is an example (the first part in the proof), where all of the three packets do not just miss a time slot. We can shift all packets so that they arrive earlier as shown in Figure 5.8(b), and the transmission delays of them become larger (from 2, 2, and 3 to 3, 3, and 4, respectively), indicating that the worst-case transmission delay never happens as Figure 5.8(a). Figure 5.8(c) is another example (the first case of the second part in the proof), where P_i just misses a time slot. If there is an unused time slot between the arriving time of P_i and the starting time of S_j, then P_i should be assigned to the unused time slot as shown in Figure 5.8(d). Figure 5.8(e) is another example (the second case of the second part in the proof), where one of the packets arriving before P_i just misses a time slot, and there is an unused time slot between the arriving time of the packet just missing a time slot and the starting time of S_j. We can shift all packets so that they arrive earlier as shown in Figure 5.8(f), and the transmission delays of the packets arriving between P_k and P_i become larger (from 2 and 3 to 3 and 4, respectively), indicating that the worst-case transmission delay never happens as Figure 5.8(e).

Given Theorem 5.2, we only need to consider a finite number of different alignments of the packet arrival pattern and the schedule pattern—they are the cases that at least one packet just misses a time slot. Here, we assume that the patterns have been repeated (duplicated) enough to A' and S' with lengths equal to the least common multiple of p and q. The worst-case transmission delay of the packet arrival pattern is

$$\max_{1\le i\le m, 1\le j\le n, 1\le k\le m} \left((s'_{j+k} - s'_j) - (a'_{i+k-1} - a'_i) \right), \tag{5.5}$$

where $a'_1, a'_2, \ldots, a'_m$ are arriving times of packets and $s'_1, s'_2, \ldots, s'_n$ are the starting times of time slots. For each (i, j, k), the equation computes the transmission delay of the $(i+k-1)$-th packet under the case that the i-th packet just misses the j-th time slot[4]. The equation can be written as

$$\max_{1\le k\le m}\left(\max_{1\le j\le n}\left(s'_{j+k}-s'_j\right)-\min_{1\le i\le m}\left(a'_{i+k-1}-a'_i\right)\right). \tag{5.6}$$

As a result, we can reduce the complexity of the computation from $O(m^2n)$ to $O(mn+m^2)$.

5.3.6 Interval Length Exploration

The latency of a packet highly depends on the length of an interval. A shorter interval results in a smaller latency of a packet, but, if the number of keys in a key chain is a constant (the memory size for storing keys), a shorter interval means that a sender needs to recompute a key chain more frequently, which increases the computational overhead. After we decide the network scheduling and compute the worst-case transmission delay, we explore the interval length of each node. For each node, there is a list of possible interval lengths, we start from the shortest one and check if the task computing a new key chain can meet its deadline which is the number of keys times the length of an interval. If it cannot meet its deadline, we will check the next possible interval length.

5.3.7 Network Scheduling Refinement

To further minimize the latency of a packet, we want keys to be released as early as possible without violating security requirement. After the interval length of each node is decided, a key can be released with the first packet in an interval. Therefore, for each sender, we check if there is any empty time slot between the starting time of a releasing interval of a sender and the first time slot assigned to a message sent by the sender. If there is such a time slot, we assign the time slot to the sender so that the sender can use the time slot to release a key. It is after the starting time of a releasing interval of a sender to satisfy the security requirement; and it is before the first time slot assigned to a message sent by the sender, so the key is released earlier, and the latency of a packet can be reduced. Furthermore, because the time slot is originally empty, it will not increase the latencies of other packets.

After this step, we can compute the worst-case latencies of time-critical paths and the objective value and check the feasibility. The mapping result, the feasibility and the objective value are returned to the simulated annealing in the outer loop.

[4]The concept here is that the densest part of the packet arrival pattern is served by the least dense part of the schedule pattern.

Table 5.1: The comparison between a non-security-aware mapping approach (its objectives are reported, but its solutions are infeasible) and our security-aware mapping algorithm for the industrial test case, where there are two optional optimization techniques resulting in four combinations. The objective is the summation of the worst-case latencies of all time-critical paths. The feasible time is the time it takes to find the first feasible solution.

		Non-Security-Aware Mapping	Security-Aware Mapping			
			No OPT1/OPT2	OPT1 Only	OPT2 Only	OPT1+OPT2
Basic SA	Objective (ms)	25,006.665	22,256.048	21,414.690	21,329.322	20,853.017
	Runtime (s)	56.435	47.046	50.725	47.767	47.862
	Feasible Time (s)	—	3.576	3.652	3.439	4.818
Accelerated SA	Objective (ms)	23,475.727	21,156.529	20,581.321	21,010.984	20,236.140
	Runtime (s)	55.695	50.441	47.963	44.070	48.065
	Feasible Time (s)	—	2.959	2.910	1.733	1.826

5.4 Experimental Results

We used the same industrial test case in Chapter 4. The architecture platform consists of 9 nodes (ECUs) which are assumed to be connected through a TDMA network (an abstraction of the Time-Triggered Ethernet or the FlexRay). The network parameters are set according to [41], while the computation time for a MAC or a key chain is scaled from [52]. The functional model consists of 41 tasks and 83 signals, and 171 paths are selected with deadlines 300 ms or 100 ms. The algorithm is implemented in C/C++. The experiments on the mapping problem are run on a 2.5-GHz processor with 4GB RAM.

We compare the results of a non-security-aware mapping approach and our security-aware algorithm. The non-security-aware mapping approach is based on the same simulated annealing core, but it does not consider any effect from the time-delayed release of keys during the mapping, *i.e.*, the latency of a packet is exactly its transmission delay because it does not need to wait a key. After the mapping is decided, we then apply the time-delayed release of keys on the design (in this step, we explore the interval of each node). On the contrary, our security-aware algorithm considers the overheads of the time-delayed release of keys from the beginning and solve the security-aware mapping problem as mentioned in Section 5.3. There are two optional optimization techniques which result in four possible combinations of the two. The first optimization technique (OPT1) is to increase R ($R > 1$) in the network scheduling, as mentioned in Section 5.3.4. The second optimization technique (OPT2) is to use empty time slots and release keys earlier in the network refinement, as mentioned in Section 5.3.7. For the simulated annealing (SA), the parameters P and Q are both set to 0.9, where Q is for the accelerated method (Accelerated SA) as mentioned in Section 5.3.2.

The results are listed in Table 5.1 where all of them are the averages of 10 runs. To have a fair comparison, we let all settings run with the same number (15,000) of iterations in the simulated annealing. The objective is the summation of the worst-case latencies of all time-critical paths. The non-security-aware mapping approach cannot find a feasible

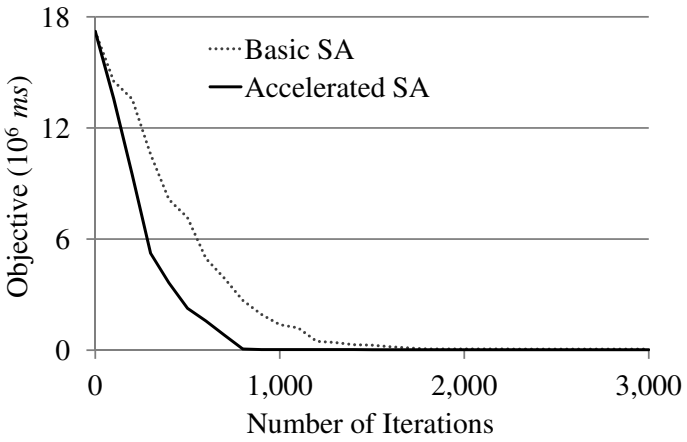


Figure 5.9: The converging behavior of the basic SA and the accelerated SA for the industrial test case. The x-axis represents the number of iterations of the simulated annealing, and the y-axis represents the objective value (10^6 *ms*) where each constraint violation contributes 10^6 to the objective value.

solution because applying the time-delayed release of keys after the mapping is decided makes some time-critical paths miss their deadlines. On the contrary, our security-aware mapping algorithm can find a feasible solution with the objective value 22,256 *ms*. If we consider the objective value of the non-security-aware mapping approach (although it is not feasible), it is 25,007 *ms*, larger than that of our security-aware mapping algorithm. This is because our security-aware mapping algorithm tends to minimize the transmission delays of asynchronous messages, which enable their senders to release keys earlier and lead to a smaller objective value. Besides, our security-aware mapping algorithm has smaller runtimes because it has a stronger force to allocate the source and target tasks of a signal to the same node and leave fewer messages on the network, which makes the runtime of network scheduling smaller.

With the two optimization techniques, the objective values are reduced to 21,415 *ms* and 21,329 *ms*, respectively. This is because, the OPT1 tries to further minimize the transmission delay of an asynchronous message, and the OPT2 tries to use empty time slots and release keys earlier. These two techniques are exactly the third and the first approaches in Section 5.3.4. By combining both of them, the objective value is further reduced to 20,853 *ms*. If the accelerated SA is applied, the design space can be explored more effectively. With the same number of iterations, the accelerated SA can find smaller objective values, compared with the basic SA. Especially, it can find a feasible solution earlier, which is because it focuses more on those tasks which play more important roles in the constraint satisfaction and the objective minimization. The converging behavior of the basic SA and the accelerated SA are illustrated in Figure 5.9. The accelerated SA converges faster than the basic SA. From the experimental results, it is difficult to tell whether OPT1 or OPT2 is more effective, but having both of them outperforms each individual. Besides, all experiments with our algorithm (using different combinations) are done within one minute. Even only

Table 5.2: The results of a large random test case.

	Security-Aware Mapping (Accelerated SA + OPT1 + OPT2)		
Signal Period Setting	1X	0.75X	0.5X
Network Utilization	0.464	0.597	0.859
Objective (ms)	20,403.161	24,714.822	45,513.222
Runtime (s)	280.823	334.415	440.395
Feasible Time (s)	21.384	29.395	60.946

considering task allocation and task priority assignment, an MILP-based approach similar to that in Chapter 4 cannot find a feasible solution in one hour. This shows the efficiency of our algorithm.

We also generate a large random test case including 500 tasks, 1,000 signals, 50 nodes, and 100 time-critical paths. We apply our security-aware mapping algorithm with the accelerated SA and the two optimization techniques (OPT1 and OPT2). In addition to algorithm scalability, we are also interested in the impact of resource availability (specifically the network utilization[5]) on the system performance and feasibility. Table 5.2 lists the objectives and runtimes under different signal periods and therefore different network utilizations (all settings are run with the same number of iterations). First of all, we can see that the algorithm scales well with the problem size because of the efficient inner loop optimization heuristics. Furthermore, as the utilization increases, the objectives and the runtimes increases dramatically. This shows the significant impact of resource availability on the system performance and feasibility when security is taken into consideration, and therefore further demonstrates the need to address security together with other metrics in an integrated formulation.

5.5 Summary

In this chapter, we present a formulation and an algorithm to consider the time-delayed release of keys and address security together with other design objectives at the design stage for TDMA-based real-time distributed systems. The algorithm optimizes the task allocation, priority assignment, network scheduling, and key-release interval length, with the consideration of the overhead and constraints from a time-delayed release of keys security mechanism. An industrial case study and a synthetic example demonstrate that our approach can effectively and efficiently explore the design space to meet all design requirements, and demonstrate the importance of considering security together with other metrics at the design stage.

[5]Here, utilization is defined as the ratio of the number of used (not scheduled) time slots to the number of total time slots.

Chapter 6

Security-Aware Design for V2V Communications

In this chapter, we further utilize our methodology to Vehicle-to-Vehicle (V2V) communications with the Dedicated Short-Range Communication (DSRC) technology. DSRC enables the development of many safety applications such as forward collision avoidance, lane change warning (blind spot warning), and left turn assist [11]. It utilizes IEEE 802.11p Wireless Access for Vehicular Environments (WAVE) [14], IEEE 802.2 [19], IEEE 1609 family (architecture [21], security services [20], network services [15], channel switching [16], electronic payment data exchange [17], identifier allocations [18]), SAE J2735 [40], SAE J2945.1 [39], and TCP/UDP/IPv6 in the United States [22]. DSRC has two options at the network and transport layers. Safety applications use the IEEE 1609.3 WAVE Short Message Protocol (WSMP) [15] which defines a WAVE Short Message and a WAVE Service Advertisement, while non-safety applications may use either the WSMP or TCP/UDP/IPv6. At the message sublayer (above the transport layer), SAE J2735 defines standard message types including a Basic Safety Message (BSM) which contains time, position, velocity, direction, size, and other important information of a vehicle.

IEEE 1609.2 [20] provides security services at the DSRC middle layers (network layer, transport layer and message sublayer). Message authentication is supported by using the Elliptic Curve Digital Signature Algorithm (ECDSA), which is an asymmetric cryptographic algorithm. When a vehicle intends to send a message, it signs the message with its private key and sends the message with its signature and certificate digest. A vehicle receiving the message then uses the public key corresponding to the private key to verify the message. The generation time of a message and the location of a vehicle are optionally included in a signed message to protect against replay attacks.

We formulate a security-aware optimization problem with consideration of both security and safety requirements, and consider the overhead of different settings of the ECDSA. The key decision variables are the sending rates and the authentication rates of BSMs which carry important information for safety applications and thus need security protections, and their sending rates and authentication rates play dominant roles in system performance [1, 22, 27].

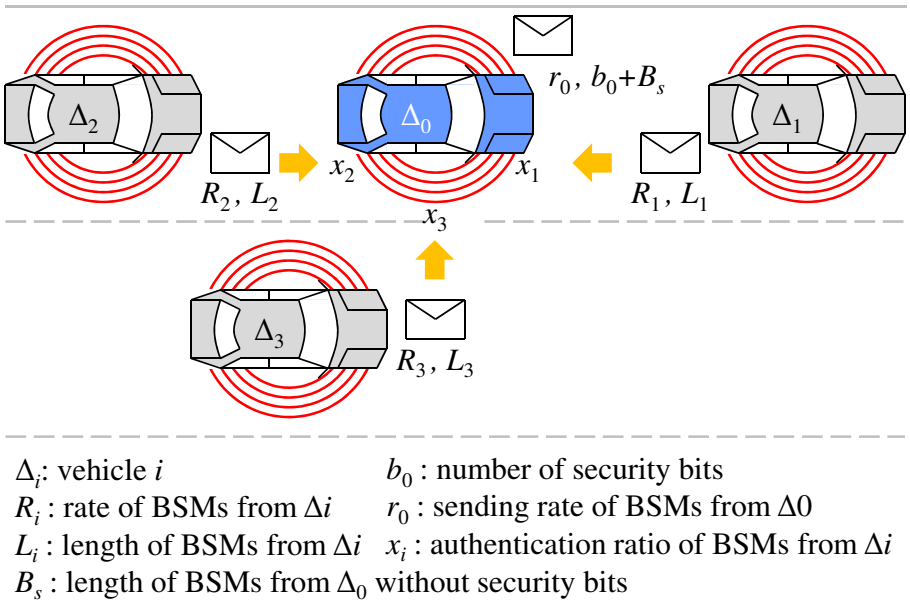


Figure 6.1: The system model.

We propose an efficient algorithm to solve the security-aware optimization problem without violating design constraints.

The chapter is organized as follows. Section 6.1 defines the formulation. Section 6.2 presents the optimization algorithm. Section 6.3 demonstrates the experimental results. Section 6.4 provides a summary of this chapter.

6.1 Formulation

The system model of the V2V communications in the security-aware optimization problem is shown in Figure 6.1. Vehicles share the DSRC channel based on the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) in the media access control layer, and each vehicle has a Wireless Safety Unit (WSU) to process BSMs. Since there is no centralized controller or vehicle, the optimization problem is solved for each vehicle separately. A vehicle Δ_0 broadcasts BSMs with its information and receives BSMs from other N vehicles $\{\Delta_1, \Delta_2, \ldots, \Delta_N\}$ with rates $\overrightarrow{R} = (R_1, R_2, \ldots, R_N)$ and lengths $\overrightarrow{L} = (L_1, L_2, \ldots, L_N)$. Since these vehicles are moving and some of them may leave the communication range of Δ_0, these values are updated at real time. The other settings are listed as follows:

- Security property: authenticity by digital signature.
- Security mechanism: Δ_0 sends BSMs with the authentication mechanism directly supported by DSRC. The authentication mechanism is abstracted to the number of additional bits to be sent with BSMs from Δ_0. On the other hand, authenticating (verifying)

all messages from other vehicles is too demanding, so a verify-on-demand authentication mechanism [1, 27] is applied to Δ_0, *i.e.*, Δ_0 only authenticates (verifies) some BSMs from other vehicles.

- Decision variables:
  1. $b_0 \in \mathbb{N}$: the number of additional security bits to be sent with BSMs from Δ_0.
  2. $r_0 \in \mathbb{R}$: the sending rate of BSMs from Δ_0.
  3. $x_i \in \mathbb{R} \cap [0, 1]$: Δ_0 authenticates (verifies) the proportion x_i of the number of BSMs from Δ_i.
- Security constraints:
  1. The number of additional security bits to be sent with BSMs from Δ_0 must be at least B_0.
  2. Δ_0 must authenticate (verify) at least the proportion X_i of the number of BSMs from Δ_i.
- Safety constraints:
  1. The sending rate of BSMs from Δ_0 must be at least R_0, which makes sure that other vehicles receive enough information.
  2. The load of the WSU of Δ_0 must not exceed a constant U_0, which controls the latency of a task on the WSU.
  3. The load of the DSRC channel must not exceed a constant U, which controls the latency of a BSM.

The problem can be interpreted by the methodology in Chapter 2, as summarized in Table 6.1, and the mathematical formulation is defined as follows:

Definition 6.1. The function u_0 to compute the load of the WSU of Δ_0 is defined as

$$u_0(r_0, b_0, \overrightarrow{x}, N, \overrightarrow{R}, \overrightarrow{L}) = r_0 T_{0,s} + \sum_{i=1}^{N} (R_i (T_{i,r} + x_i T_{i,v})), \tag{6.1}$$

where $T_{0,s}$ is the processing time for Δ_0 to send and sign a BSM, $T_{i,r}$ is the processing time for Δ_0 to receive a BSM from Δ_i, and $T_{i,v}$ is the processing time for Δ_0 to verify a BSM from Δ_i. These values may depend on b_0 and $\overrightarrow{L}$.

Definition 6.2. The function u to compute the load of the DSRC channel is defined as

$$u(r_0, b_0, N, \overrightarrow{R}, \overrightarrow{L}) = r_0 (B_s + b_0) + \sum_{i=1}^{N} (R_i L_i), \tag{6.2}$$

where B_s is the length of a BSM without additional security bits.

Table 6.1: The V2V security-aware optimization problem interpreted by the security-aware design methodology.

		V2V Security-Aware Optimization Problem
$\mathcal{F}$	M_F	tasks sending, receiving, signing, and verifying BSMs
	Q_F	authenticity by digital signature
	R_F	security constraints (1) and (2)
	C_F	safety constraints (1)–(3)
	O_F	Weighted numbers of verified BSMs
$\mathcal{A}$	A_A	WSU and DSRC
	V_A	—
$\mathcal{S}$	S_S	digital signature supported by DSRC
	R_S	—
	V_S	—
$X_{F,A}$		r_0
X_S		b_0, x_i

Definition 6.3. Given N, $\overrightarrow{R}$, $\overrightarrow{L}$, decide $r \in \mathbb{R}$, $b \in \mathbb{N}$, and $\overrightarrow{x} \in \{\mathbb{R} \cap [0,1]\}^N$ such that

$$b_0 \geq B_0, \tag{6.3}$$
$$\forall i, \quad x_i \geq X_i, \tag{6.4}$$
$$r_0 \geq R_0, \tag{6.5}$$
$$u_0(r_0, b_0, \overrightarrow{x}, N, \overrightarrow{R}, \overrightarrow{L}) \leq U_0, \tag{6.6}$$
$$u(r_0, b_0, N, \overrightarrow{R}, \overrightarrow{L}) \leq U, \tag{6.7}$$

and maximize the weighted summation of the numbers of verified BSMs from other vehicles (per time unit), *i.e.*,

$$\text{maximize} \quad \sum_{i=1}^{N} W_i(R_i x_i), \tag{6.8}$$

where W_i is the weight, depending on the criticality of BSMs from Δ_i.

W_i is a function of the velocity of Δ_i and the distance between Δ_0 and Δ_i. It also depends on the application. For example, if the application is forward collision avoidance, then the BSMs from the vehicle in front of Δ_0 are extremely critical, so a larger W_i should be assigned to it. Note that, if $W_i = 1$ for all i, then the objective becomes to maximize the number of BSMs from other vehicles (per time unit). The latency of a task or a BSM can also be considered in the formulation. However, it highly depends on the load of the WSU or the DSRC channel, so we focus on the load constraints here. If the latency is

Algorithm 3 Algorithm for security-aware optimization.

1: $\delta = U_0 - R_0 T_{0,s} - \sum_{i=1}^{N}(R_i(T_{i,r} + T_{i,v}X_i))$;
2: **if** $R_0(B_s + B_0) + \sum_{i=1}^{N}(R_i L_i) > U$ **or** $\delta < 0$ **then**
3: Return "infeasible";
4: **end if**
5: $b_0 = B_0$; $r_0 = R_0$; $\forall i, x_i = X_i$;
6: Sort $\frac{W_i}{T_{i,v}}$ in descending order;
7: **for** each index set $\{i_1, i_2, \ldots, i_M\}$ where $\frac{W_{i_j}}{T_{i_j,v}} = \frac{W_{i_k}}{T_{i_k,v}}$ for all $1 \le j, k \le M$ **do**
8: **if** $\sum_{j=1}^{M} R_{i_j} T_{i_j,v}(1 - X_{i_j}) < \delta$ **then**
9: **for** each k, $1 \le k \le M$ **do**
10: $x_{i_k} = 1$;
11: **end for**
12: $\delta = \delta - \sum_{j=1}^{M} R_{i_j} T_{i_j,v}(1 - X_{i_j})$;
13: **else**
14: **for** each k, $1 \le k \le M$ **do**
15: $x_{i_k} = x_{i_k} + \frac{\delta(1-X_{i_k})}{\sum_{j=1}^{M} R_{i_j} T_{i_j,v}(1-X_{i_j})}$;
16: **end for**
17: $\delta = 0$;
18: Return $(r_0, b_0, \overrightarrow{x})$;
19: **end if**
20: **end for**
21: Return $(r_0, b_0, \overrightarrow{x})$;

considered, due to the hidden node problem in wireless communications and the real-time changes of N, $\overrightarrow{R}$, $\overrightarrow{L}$, it can only be estimated. Any model for the latency computation can be applied, depending on the complexity and accuracy of the model. Lastly, in this formulation, architecture selection is not considered since DSRC is designated for V2V communications, while security mechanism selection is implied by the abstraction of the number of additional bits sent with BSMs and the processing times of signing and verifying BSMs. Note that the optimization problem can be solved at real time or at the design stage with different input parameters. If it is solved at the design stage, designers can construct a lookup table to save the solutions, and a vehicle can check the lookup table for solutions at real time without solving the problem again.

6.2 Algorithm

We propose an efficient algorithm (Algorithm 3) to solve the security-aware optimization problem. It is optimal for the problem, and the overall time complexity is $O(N \log N)$. In Lines 1–4, the algorithm first checks the two load constraints, Equations (6.6) and (6.7),

with the minimal possible values of r_0, b_0, and x_i, which generate the minimal loads for the WSU or the DSRC channel. If any of them is not satisfied, the problem is infeasible. In Line 5, the algorithm assigns initial values to r_0, b_0, and x_i. In Line 6, the algorithm sorts $\frac{W_i}{T_{i,v}}$. In Lines 7–20, the algorithm applies a greedy approach starting from the maximal $\frac{W_i}{T_{i,v}}$. If the corresponding $\frac{W_i}{T_{i,v}}$'s are the same for a set of indices, instead of just selecting one of them, the algorithm considers all of them at the same time. For each index in the set, without violating Equation (6.6), the algorithm assigns the maximal value to x_i (Line 10) or increases x_i by a constant ratio of the difference to its maximal value (Line 15). After that, the value of δ is updated (Lines 12 and 17) so that it is always the difference between the two sides of Equation (6.6). If δ becomes 0, meaning that no x_i can be increased any more, or all x_i's are assigned to their maximal values, the algorithm returns the solution (Lines 18 and 21).

Theorem 6.1. *Algorithm 3 is optimal for the problem in Definition 6.3.*

Proof. The algorithm first assigns the minimal possible values to r_0, b_0, and x_i, which generate the minimal but required loads for the WSU or the DSRC channel. Next, the algorithm increases the value of x_i. To prove the optimality, we can replace x_i by $\frac{x'_i}{R_i T_{i,v}} + X_i$, and the problem can be transformed to maximize $\sum_{i=1}^{N} A'_i x'_i + B'$, subject to $\sum_{i=1}^{N} x'_i \leq C'$ and $0 \leq x'_i \leq X'_i$, where $x'_i = R_i T_{i,v}(x_i - X_i)$, $X'_i = R_i T_{i,v}(1 - X_i)$, $A'_i = \frac{W_i}{T_{i,v}}$, $B' = \sum_{i=1}^{N}(W_i R_i X_i)$, and $C' = U_0 - R_0 T_{0,s} - \sum_{i=1}^{N}(R_i(T_{i,r} + T_{i,v} X_i))$.

X'_i, A'_i, B', and C' are all constants, and the coefficient of each x'_i in constraints is 1, so an approach which greedily increases the value of an x'_i with the largest A'_i can be applied. This is because, among increasing each x'_i by the same amount which has the same effect on constraints, the selection of an x'_i with the largest A'_i is the most effective in maximizing the objective. Similarly, if the corresponding A'_i's are the same for several x'_i's, increasing the values of these x'_i's without violating constraints is optimal, which is exactly what Algorithm 3 does in Lines 7–21. Lastly, because of the checks in Lines 1–4 and updates of δ, the algorithm returns a feasible solution if and only if the problem has a feasible solution. □

Theorem 6.2. *The overall time complexity of Algorithm 3 is $O(N \log N)$.*

Proof. The time complexity of Lines 1–5 is $O(N)$ for the summations. The sorting in Line 6 takes $O(N \log N)$. The time complexity of Lines 7–20 is $O(N)$ since they are only executed once for each index i. Therefore, the overall time complexity of Algorithm 3 is $O(N \log N)$. □

6.3 Experimental Results

If not specified otherwise, the parameters used in the experiments are listed in Table 6.2. Especially, we assume that a vehicle has a 400-MHz WSU, and the time that it needs to sign

Table 6.2: The parameters in the experiments.

Parameter	Unit	ECDSA-224	ECDSA-256
B_0	bit	$(56+8+6)\times 8$	$(64+8+6)\times 8$
R_0	$1/ms$	0.002	
X_i	—	0.05	
U_0	1	0.8	
U	$kbps$	300	
R_i	$1/ms$	0.002	
L_i	bit	$(200+70)\times 8$	$(200+78)\times 8$
W_i	—	1	
$T_{0,s}$	ms	5.9+1.0	7.6
$T_{i,r}$	ms	1.0	1.0
$T_{i,v}$	ms	17.8	26.5
B_s	bit	200×8	

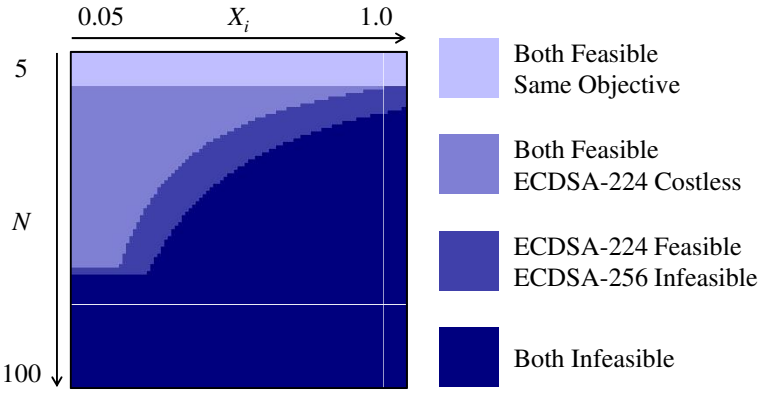


Figure 6.2: The selection between the ECDSA-224 and the ECDSA-256.

or verify a BSM is given [1]. The values of B_0 and L_i can be calculated [27]. The load of the WSU is bounded by 0.8, and the load of the DSRC channel is bounded by 300 *kbps*, 60% of the full load [22]. The algorithm is implemented in the C language, and the experiments are run on a 2.5-GHz processor. There are about 50,000 instances of the optimization problem solved in 1 second, showing the efficiency of the algorithm and making it possible to be used at real time, even with a slower processor, such as the WSU in a previous work [1]. There are three settings experimented and discussed in the following sections.

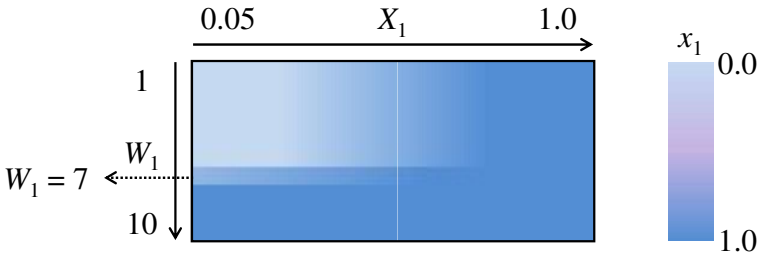


Figure 6.3: The verification percentage x_i under different W_i and X_i.

6.3.1 Selection Between ECDSA-224 and ECDSA-256

We compare the security mechanisms, the ECDSA-224 and the ECDSA-256, which are directly supported by DSRC. In this setting, N (the number of vehicles except Δ_0) ranges from 5 to 100, and each X_i (the minimum verification proportion) ranges from 0.05 to 1.0. The results are shown in Figure 6.2. When the load of the WSU is very low, both of the ECDSA-224 and the ECDSA-256 are feasible with the same objective value because the WSU verifies all BSMs from all vehicles with either the ECDSA-224 or the ECDSA-256. As N and X_i increase, more BSMs are generated, and a higher proportion of the number of BSMs needs to be verified, causing higher loads of the WSU. As a result, the WSU cannot verify all BSMs, and the ECDSA-224 has a larger (better) objective value due to its smaller computational overhead. As N and X_i further increase, the ECDSA-256 first becomes infeasible, and then the ECDSA-224 becomes infeasible as well. This experiment demonstrates that, even in some non-extreme cases, verifying all BSMs from other vehicles is too demanding, so the verify-on-demand authentication mechanism [1, 27] is necessary. Besides, when the computational overhead is a concern, the ECDSA-224 is a better choice, which is also mentioned in a previous work [22].

6.3.2 Changing Weight and Minimum Verification Proportion

We consider the scenario that BSMs from a specified vehicle are especially critical. For example, if the application is forward collision avoidance, the BSMs from the vehicle in front of Δ_0 are extremely critical. To test this scenario, the ECDSA-224 is selected, N is set to 51, W_1 ranges from 1 to 10, X_1 ranges from 0.05 to 1.0, and $W_i = \left\lfloor \frac{i-2}{5} + 1 \right\rfloor$ for $2 \leq i \leq 51$. Note that W_i is the weight in the objective function, and X_i is the minimum verification proportion. The results are shown in Figure 6.3. With this setting, the load of the WSU is fixed, but the WSU cannot afford to verify all BSMs from all vehicles. Recall that, after the required proportion X_i is assigned to x_i, Algorithm 3 further increases x_i by a greedy approach. When W_1 is small ($W_1 < 7$ in this case), x_1 is sorted behind and not increased because δ (the difference between the two sides of Equation (6.6)) becomes 0 before Algorithm 3 considers x_i. On the other hand, when W_1 is large ($W_1 > 7$ in this case), x_1 is sorted to front and increased to 1 because δ is still non-negative after that. When W_1 is on

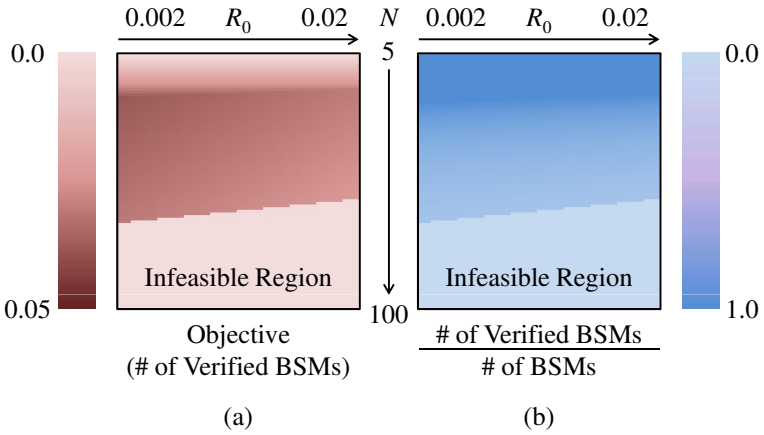


Figure 6.4: The trade-off between security and BSM sending rate.

the boundary ($W_1 = 7$ in this case), x_1 is increased but not to 1, and the amount depends on all X_i's where $W_1 = W_i$ because Algorithm 3 considers all of them at the same time, not just selects one of them. These results show that assigning appropriate values to W_i and X_i leads Algorithm 3 to emphasize some important BSMs.

6.3.3 Trade-Off Between Security and BSM Sending Rate

We experiment on the trade-off between security and BSM sending rate. The safety metric being considered here has a relatively narrow scope, which focuses on the amount of information being exchanged that can be used for safety application. In a broader sense, safety also depends on security, in which case a single metric may be defined. In this setting, N ranges from 5 to 100, and R_0 (the minimum sending rate) ranges from 0.002 to 0.02. The results are shown in Figure 6.4, where 0.0 means that there is no feasible solution. Figure 6.4(a) shows the objective which is exactly the number of verified BSMs (per ms) since $W_i = 1$ for all i, while Figure 6.4(b) shows the ratio of the number of verified BSMs to the total number of BSMs. The objective and the ratio are "security measurements" because larger values of them indicate that more BSMs are verified. On the other hand, R_0 is a "safety measurement" because a larger value of R_0 means more information provided to other vehicles. Note that, in this setting, the total number of BSMs is fixed if N is fixed.

In Figure 6.4(a) and (b), if N is fixed, the objective and the ratio (security measurements) decrease as R_0 (safety measurement) increases. It demonstrates the trade-off between security and BSM sending rate. Besides, if R_0 is fixed, the ratio also decreases as N increases in Figure 6.4(b), which shows the trade-off between security and system scalablility. However, the objective does not always decrease as N increases. This is because, when N and R_0 are small, all BSMs can be verified, and thus the number of verified BSMs increases as N increases. There is also trade-off between sending and receiving BSMs. It can be observed by the boundary between a feasible region and an infeasible region in Figure 6.4, where, as R_0

increases, the maximal feasible N decreases. These results indicate that considering security induces more design challenges, and systematic approaches and design tools are crucial to constraint satisfaction and design optimization.

6.4 Summary

In this chapter, we apply the methodology in Chapter 2 to V2V communications through DSRC. The key decision variables are the sending rates and the authentication rates of BSMs which carry important information for safety applications and thus need security protections, and their sending rates and authentication rates play dominant roles in system performance. Experimental results demonstrate the efficiency of our algorithm which solves the security-aware optimization problem without violating design constraints.

Chapter 7

Conclusions and Future Work

In this thesis, to address security with limited resources and strict constraints in embedded systems, we proposed a general security-aware design methodology which considers security together with other design constraints at design stages. The methodology is based on Platform-Based Design [44], where a functional model and an architectural platform are initially captured separately and then brought together through a mapping process. During mapping, the functional model is implemented on the architectural platform, and constraints and objectives are satisfied and optimized, respectively. Our methodology is different from the traditional mapping process because it not only maps functional models to architectural platforms but also explores security mechanism selection and architecture selection.

We then focused on the security issues for automotive systems as they represent many of the common challenges in embedded systems. We studied security for in-vehicle communications and presented security mechanisms for the CAN protocol, which is a very representative asynchronous protocol and currently the most used in-vehicle communication protocol. Based on the security mechanisms, we proposed an MILP formulation and an MILP-based algorithm to explore task allocation, signal packing, MAC sharing, and priority assignment and meet both security and safety constraints. Besides the CAN protocol, we also considered a TDMA-based protocol, which is a very representative synchronous protocol and an abstraction of many existing protocols. The time-delayed release of keys [2, 34, 35, 52] was applied as the security mechanism, and an algorithm that combines a simulated annealing approach with a set of efficient optimization heuristics was developed to solve a security-aware mapping problem for TDMA-based systems. Lastly, we applied our methodology to V2V communications with the DSRC technology. We formulated a security-aware optimization problem and proposed an efficient algorithm to solve the security-aware optimization problem.

Experimental results showed that our approaches can effectively and efficiently explore design spaces and satisfy all design constraints at the design stages. They also demonstrated that security must be considered at initial design stages; otherwise, it is very difficult and sometimes impossible to add security after initial design stages without violating other system constraints.

Based on the contributions in this thesis, potential future directions include:

- Different applications: different types of attacks have been identified in aircraft systems [6, 43], global positioning systems [8, 55], medical devices [10, 28], and smart grids [23, 29]. Although some security mechanisms have been proposed for those applications, security and its impact on system design are not considered at their design stages to guarantee that all design constraints are satisfied. Different applications may have different bottlenecks for security implementation. Some challenges, such as power consumption for medical devices and scalability for smart grids, have been pointed out, but there is still no rigorous approach to address these challenges at the design stages. Our security-aware design methodology can be applied to deal with the security-aware design problems for those applications.

- Different security properties: authentication is the main focus in Chapters 3–6, while different applications may have have different properties to be fulfilled. Availability is a concern for most applications, and it usually needs some hardware protection. On the other hand, confidentiality is clearly a concern for medical devices and smart grids. To address these security properties, our security-aware design can also be applied to deal with them.

- Design and analysis over DSRC: DSRC provides a very good platform for many applications of automotive systems. Along with security, other metrics, such as safety, energy efficiency, and congestion avoidance, and any of their combinations are important and intriguing topics in design and analysis problems. Especially, there is enormous information from vehicles and infrastructures, so data mining techniques are needed to determine which message is relevant. Besides, due to the moving nature of automotive systems, expectation maximization techniques are also needed to predict the changing of surrounding environment. Those machine learning techniques are expected to work closely with design and analysis algorithms.

Bibliography

[1] F. Ahmed-Zaid, F. Bai, S. Bai, C. Basnayake, B. Bellur, S. Brovold, G. Brown, L. Caminiti, D. Cunningham, H. Elzein, K. Hong, J. Ivan, D. Jiang, J. Kenney, H. Krishnan, J. Lovell, M. Maile, D. Masselink, E. McGlohon, P. Mudalige, Z. Popovic, V. Rai, J. Stinnett, L. Tellis, K. Tirey, and S. VanSickle. Vehicle safety communications—applications (VSC-A). *Report No. DOT HS 811 492A*, September 2011.

[2] R. Anderson, F. Bergadano, B. Crispo, J.-H. Lee, C. Manifavas, and R. Needham. A new family of authentication protocols. *ACM SIGOPS Operating Systems Review*, 32(4):9–20, October 1998.

[3] Bosch. CAN specification (Version 2.0). 1991.

[4] Bosch. CAN with flexible data-rate white paper, (Version 1.1). 2011.

[5] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno. Comprehensive experimental analyses of automotive attack surfaces. In *USENIX Conference on Security*, pages 6–6, 2011.

[6] R. De Cerchio and C. Riley. Aircraft systems cyber security. In *Integrated Communications, Navigation and Surveillance Conference (ICNS)*, pages 1–12, April 2012.

[7] FlexRay Consortium. FlexRay communications system protocol specification (Version 3.0.1). October 2010.

[8] S. Gong, Z. Zhang, M. Trinkle, A. D. Dimitrovski, and H. Li. GPS spoofing based time stamp attack on real time wide area monitoring in smart grid. In *IEEE Third International Conference on Smart Grid Communications (SmartGridComm)*, pages 300–305, November 2012.

[9] B. Groza, S. Murvay, A. Van Herrewege, and I. Verbauwhede. LiBrA-CAN: a lightweight broadcast authentication protocol for Controller Area Networks. In *International Conference on Cryptology and Network Security*, pages 185–200, December 2012.

[10] D. Halperin, T. S. Heydt-Benjamin, B. Ransford, S. S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. H. Maisel. Pacemakers and implantable cardiac defibrillators:

software radio attacks and zero-power defenses. In *IEEE Symposium on Security and Privacy (SP)*, pages 129–142, May 2008.

[11] J. Harding, G. Powell, R. Yoon, J. Fikentscher, C. Doyle, D. Sade, M. Lukuc, J. Simons, and J. Wang. Vehicle-to-vehicle communications: readiness of V2V technology for application. *Report No. DOT HS 812 014*, August 2014.

[12] T. Hoppe, S. Kiltz, and J. Dittmann. Security threats to automotive CAN networks—practical examples and selected short-term countermeasures. In *International Conference on Computer Safety, Reliability, and Security*, pages 235–248, 2008.

[13] ICAO. Manual for the ATN using IPS standards and protocols (doc 9896). September 2010.

[14] IEEE. IEEE standard for information technology—local and metropolitan area networks—specific requirements—part 11: wireless LAN medium access control (MAC) and physical layer (PHY) specifications amendment 6: wireless access in vehicular environments. *IEEE Std 802.11p-2010 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, IEEE Std 802.11n-2009, and IEEE Std 802.11w-2009)*, July 2010.

[15] IEEE. IEEE standard for wireless access in vehicular environments (WAVE)—networking services—redline. *IEEE Std 1609.3-2010 (Revision of IEEE Std 1609.3-2007)—Redline*, December 2010.

[16] IEEE. IEEE standard for wireless access in vehicular environments (WAVE)—multi-channel operation. *IEEE Std 1609.4-2010 (Revision of IEEE Std 1609.4-2006)*, February 2011.

[17] IEEE. IEEE standard for wireless access in vehicular environments (WAVE)—over-the-air electronic payment data exchange protocol for intelligent transportation systems (ITS). *IEEE Std 1609.11-2010*, January 2011.

[18] IEEE. IEEE standard for wireless access in vehicular environments (WAVE)—identifier allocations. *IEEE Std 1609.12-2012*, September 2012.

[19] IEEE. IEEE standard for information technology—telecommunications and information exchange between systems—local and metropolitan area networks—specific requirements—part 2: logical link control. *ISO 8802-2 IEEE 802.2, First Edition 1989-12-31 (Revision of IEEE Std 802.2-1985)*, February 2013.

[20] IEEE. IEEE standard for wireless access in vehicular environments security services for applications and management messages. *IEEE Std 1609.2-2013 (Revision of IEEE Std 1609.2-2006)*, April 2013.

[21] IEEE. IEEE guide for wireless access in vehicular environments (WAVE)—architecture. *IEEE Std 1609.0-2013*, March 2014.

[22] J. B. Kenney. Dedicated short-range communications (DSRC) standards in the United States. *Proceedings of the IEEE*, 99(7):1162–1182, July 2011.

[23] H. Khurana, M. Hadley, N. Lu, and D. A. Frincke. Smart-grid security issues. *IEEE Security & Privacy*, 8(1):81–85, January 2010.

[24] P. Kleberger, T. Olovsson, and E. Jonsson. Security aspects of the in-vehicle network in the connected car. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 528–533, June 2011.

[25] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage. Experimental security analysis of a modern automobile. In *IEEE Symposium on Security and Privacy (SP)*, pages 447–462, May 2010.

[26] F. Koushanfar, A.-R. Sadeghi, and H. Seudie. EDA for secure and dependable cybercars: challenges and opportunities. In *ACM/IEEE Design Automation Conference (DAC)*, pages 220–228, June 2012.

[27] H. Krishnan and A. Weimerskirch. "verify-on-demand"—a practical and scalable approach for broadcast authentication in vehicle-to-vehicle communication. *SAE International Journal of Passenger Cars—Mechanical Systems*, 4(1):536–546, June 2011.

[28] C. Li, A. Raghunathan, and N. K. Jha. Hijacking an insulin pump: security attacks and defenses for a diabetes therapy system. In *IEEE International Conference on e-Health Networking Applications and Services (Healthcom)*, pages 150–156, June 2011.

[29] P. McDaniel and S. McLaughlin. Security and privacy challenges in the smart grid. *IEEE Security & Privacy*, 7(3):75–77, May 2009.

[30] A. R. Metke and R. L. Ekl. Security technology for smart grid networks. *IEEE Transactions on Smart Grid*, 1(1):99–107, June 2010.

[31] M. Di Natale, H. Zeng, P. Giusto, and A. Ghosal. Worst-case time analysis of can messages. In *Understanding and Using the Controller Area Network Communication Protocol*, pages 43–65. Springer, 2012.

[32] D. K. Nilsson, U. E. Larson, and E. Jonsson. Efficient in-vehicle delayed data authentication based on compound message authentication codes. In *IEEE Vehicular Technology Conference (VTC)*, pages 1–5, September 2008.

[33] OSEK. OSEK/VDX OS specification, (Version 2.2.3). 2006.

[34] A. Perrig, R. Canetti, D. Song, and J. D. Tygar. Efficient and secure source authentication for multicast. In *Network and Distributed System Security Symposium*, pages 35–46, 2001.

[35] A. Perrig, R. Canetti, J. D. Tygar, and D. Song. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy (SP)*, pages 56–73, 2000.

[36] I. Rouf, R. Miller, H. Mustafa, T. Taylor, S. Oh, W. Xu, M. Gruteser, W. Trappe, and I. Seskar. Security and privacy vulnerabilities of in-car wireless networks: a tire pressure monitoring system case study. In *USENIX Conference on Security*, pages 21–21, 2010.

[37] RTCA. Minimum aviation system performance standards for automatic dependent surveillance—broadcast (ADS-B). *RTCA DO-242A*, June 2002.

[38] U. Ruhrmair and M. van Dijk. PUFs in security protocols: attack models and security evaluations. In *IEEE Symposium on Security and Privacy (SP)*, pages 286–300, May 2013.

[39] SAE. Dedicated short range communication (DSRC) minimum performance requirements. *SAE Standard J2945 WIP*.

[40] SAE. Dedicated short range communications (DSRC) message set dictionary. *SAE Standard J2735*, November 2009.

[41] SAE. Time-Triggered Ethernet. *SAE Standard AS6802*, November 2011.

[42] SAE. TTP communication protocol. *SAE Standard AS6003*, February 2011.

[43] K. Sampigethaya, R. Poovendran, S. Shetty, T. Davis, and C. Royalty. Future e-enabled aircraft communications and security: the next 20 years and beyond. *Proceedings of the IEEE*, 99(11):2040–2055, November 2011.

[44] A. Sangiovanni-Vincentelli. Quo vadis, SLD? reasoning about the trends and challenges of system level design. *Proceedings of the IEEE*, 95(3):467–506, March 2007.

[45] S. Seifert and R. Obermaisser. Secure automotive gateway — secure communication for future cars. In *IEEE International Conference on Industrial Informatics (INDIN)*, pages 213–220, July 2014.

[46] G. E. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. In *ACM/IEEE Design Automation Conference (DAC)*, pages 9–14, June 2007.

[47] C. Szilagyi and P. Koopman. A flexible approach to embedded network multicast authentication. In *Workshop on Embedded Systems Security*, 2008.

[48] C. Szilagyi and P. Koopman. Flexible multicast authentication for time-triggered embedded control network applications. In *IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 165–174, 2009.

[49] C. Szilagyi and P. Koopman. Low cost multicast authentication via validity voting in time-triggered embedded control networks. In *Workshop on Embedded Systems Security*, 2010.

[50] A. Van Herrewege, D. Singelee, and I. Verbauwhede. CANAuth—a simple, backward compatible broadcast authentication protocol for CAN bus. In *Workshop on Embedded Security in Cars*, 2011.

[51] A. Wasicek, P. Derler, and E. A. Lee. Aspect-oriented modeling of attacks in automotive cyber-physical systems. In *ACM/IEEE Design Automation Conference (DAC)*, pages 21:1–21:6, June 2014.

[52] A. Wasicek, C. El-Salloum, and H. Kopetz. Authentication in time-triggered systems using time-delayed release of keys. In *IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC)*, pages 31–39, March 2011.

[53] M. Wolf and T. Gendrullis. Design, implementation, and evaluation of a vehicular hardware security module. In *Proceedings of the 14th International Conference on Information Security and Cryptology*, pages 302–318. Springer-Verlag, 2012.

[54] M.-D. Yu and S. Devadas. Secure and robust error correction for physical unclonable functions. *IEEE Design Test of Computers*, 27(1):48–65, January 2010.

[55] Q. Zeng, H. Li, and L. Qian. GPS spoofing attack on time synchronization in wireless networks and detection scheme design. In *Military Communications Conference (MILCOM)*, pages 1–5, October 2012.

[56] Q. Zhu, Y. Yang, M. Di Natale, E. Scholte, and A. Sangiovanni-Vincentelli. Optimizing the software architecture for extensibility in hard real-time distributed systems. *IEEE Transactions on Industrial Informatics*, 6(4):621–636, November 2010.

[57] T. Ziermann, S. Wildermann, and J. Teich. CAN+: A new backward-compatible controller area network (CAN) protocol with up to 16x higher data rates. In *ACM/IEEE Design, Automation Test in Europe (DATE)*, pages 1088–1093, April 2009.