博 士 論 文

# Anomaly Detection Technology for In-vehicle Networks

Jun YAJIMA
矢嶋 純

情報セキュリティ大学院大学
情報セキュリティ研究科
情報セキュリティ専攻

2021年3月

# Contents

# Chapter 1

# Introduction

Modern cars are electronically controlled by in-vehicle networks while they are running; electronic controls are also used to open and close their windows, etc. Furthermore, the number of cars connected to external networks such as the Internet are increasing. Such vehicles are called connected cars. In particular, an external network connection is needed for self-driving cars. Cyberattacks against connected cars is a growing problem. In 2015, a cyberattack against actual vehicles was demonstrated [1]. In this attack, researchers exploited vulnerabilities of the in-vehicle unit, whereby they could control cars remotely. After that, 1.4 million cars were recalled by the manufacturer of the car models that were attacked. This was the first ever recall for cybersecurity reasons.

Given such background, many organizations have decided to study the security of vehicles and have standardized technologies. In particular, the UNECE

World Forum for Harmonization of Vehicle Regulations (WP.29) of the United Nations Economic Commission for Europe (UNECE) has standardized vehicle regulations for protecting against cyberattacks [2]. To comply with these regulations, a cyber security management system and software update management system are mandatory. The Society of Automotive Engineers (SAE) is standardizing ISO/SAE 21434 [3] that regulates vehicle life-cycle management [4]. Satisfying this standard is needed to meet the regulations of WP.29. In the regulation of functional safety ISO 26262 [5], support for cybersecurity is mandatory. Other organizations have standardized regulations supporting cybersecurity for vehicles. Although their standards indicate that security should be considered, no specific technology has been standardized.

We think that cyberattacks against cars are serious problems and that countermeasures are needed to combat them. In order to secure cars, we initially examined the various security strategies for a general IT environment, i.e., multi-layered security [6][7], defense in depth [8], and security management cycle [9]. After that, we examined security strategies specifically for vehicles. As a result, we determined that attack detection constitutes one of the most crucial issues because conventional IT technologies are inadequate for this purpose in an in-vehicle network. Research into attack detection techniques for vehicles is hence a major research topic. The findings indicate that, once detected, attacks can be stopped by shutting off connections to external networks and parts of the in-vehicle networks.

The attacks against certain vulnerabilities can be analyzed by utilizing detection logs, and security patches can be developed and sent to cars of the same type. We think that these steps, namely, protection, detection, recovery, and update, constitute a security cycle for in-vehicle units.

In this dissertation, we discuss cyberattack detection technology for in-vehicle networks. There are two types of attacks against vehicles: direct attacks and indirect attacks. In direct attacks, attackers connect the attack device directly to the in-vehicle network. On the other hand, in indirect attacks, attackers abuse the vulnerabilities of in-vehicle units without entering the car. We think that attackers who would try to conduct indirect attacks are more numerous than attackers who would try to conduct direct attacks, because constructing an attack environment for indirect attacks is easier than that for direct attacks. Therefore, countermeasures against indirect attacks should be prioritized over countermeasures against direct attacks.

Regarding methods of detecting indirect attacks, there are two typical strategies: one that utilizes reception periodicity and another that analyzes messaging behavior. We consider that detection technologies that utilize the reception period are especially important because many of the messages on in-vehicle networks that are used for vehicle control are sent periodically. Therefore, our aim is to construct a detection method that utilizes the message reception period. The first method we developed is a detection technique for periodic messages. Many of the

existing techniques for periodic messages cover only perfectly periodic messages [10][11]. However, we found that the transmission periods of many messages are often biased in some way and that a high-accuracy detection is needed. The first method thus covers not only perfectly periodic but also quasi-periodic messages whose transmission period may be biased.

However, we cannot ignore event-based and non-periodic messages. The second and third methods we propose are for these event-based and non-periodic messages. These methods are not detection methods in themselves but are helpful for constructing detection methods utilizing messaging behavior. In particular, to construct detection method utilizing messaging behavior, we should analyze the behavior of normal messages. Our second method estimates the formats of the payloads of messages of in-vehicle networks. The third method analyzes the relationship between multiple data in the payloads of received messages.

As mentioned above, the second method estimates the format of the payload of the received messages. A typical protocol for in-vehicle networks enables data to be stored as the payload of a message. In many cases, multiple data are stored in a payload. The format shows the type of the data (type), the start position in the payload for the data (position), and the length of the data (length). If the detectors know the format, they can easily construct detection techniques. Therefore, knowing the format is very important. Typically, this information is not published, though it may be known to the car manufacturer. In particular, car manufactur-

ers may know part of the format of the payload of the Controller Area Network (CAN), but don't know the behavior of that data in detail. If they don't know the behavior of the data, they can construct detection methods that contain information on normal behaviors. If they know the behavior of the data, they can confirm that whether their information is correct or not. Moreover, car manufactureres may not know the format of messages sent from equipments of third party maker that may be installed after released. The second method can be used to such messages. The existing format estimation methods are detailed in [12], [13], [14]. We propose a method that can estimate formats more precisely than the existing methods can.

The third method finds relations between data in the payloads of multiple messages. Such "relations" indicate the data that have the same number of changes at almost the same timing, or same number of identical values at almost the same timing. The found relations can then be utilized to construct detection techniques. The usefulness of knowing relations between multiple messages has not been pointed out in the previous research. Finally, by combining methods 2 and 3, we can construct methods for dealing with various kinds of non-periodic messages.

The above discussion pertains to rules-based detection. However, machine-learning-based detection is also being actively researched. In these methods, network logs are collected and used for training a machine-learning model. By training a machine-learning model with many logs, the manner of detection can be

designed automatically. We hypothesized that the detection accuracy when using the correct format is higher than when using incorrect formats, and we confirmed this to be the case in a computer experiment. Moreover, we found that format estimation is also important for machine-learning-based detections.

Our methods can accurately detect attacks for all types of messages. Method 1, based on message periodicity, can detect attacks with almost no false positives or negatives for perfectly periodic and quasi-periodic messages under some assumptions. This level of accuracy has not been achieved by any of the existing methods based on the same strategy. The assumptions are natural ones for in-vehicle networks. Method 2 can be used to build a messaging-behavior-based detection. It reveals the format of messages that are useful for detecting attacks, and it reveals them with higher accuracy than the existing methods. The method 3 is also helpful for building messaging-behavior-based detections. It reveals the relations between data in different messages that are also useful for detecting attacks. The strategy behind our methods is new; namely, no other methods focus on relations between data in multiple messages.

Our techniques can realize highly accurate detection for the following networks.

1. There are messages sent periodically.

2. Data storage location is the same in the same meaning packets.

Proposal 1 can be used for 1. Proposals 2 and 3 can be used for 2. The protocols used in the in-vehicle networks are quite different from those used in the IT environments, such as the absence of 'From' or 'To'. The conditions 1 and 2 are exactly the in-vehicle network itself. Our proposed methods enable high accurate detection on the in-vehicle networks, which has not been realized by the attack detection techniques for IT environments. We think that our techniques can be applied to protocols used in other environments that satisfy the above 1 and 2.

This dissertation is structured as follows. Cyberattacks against vehicles are described in Chapter 2. In Chapter 3, we discuss security strategies for general IT environments. Our strategy for securing vehicles against cyberattacks is described in Chapter 4. We explain in-vehicle networks and its typical protocol in Chapter 5. Attack detection strategies are shown in Chapter 6. After that, our first proposal, "Anomaly Detection by Cumulative Sum (ADCS)," is explained in Chapter 7. Our proposal for format estimation, "Single Message Format Estimation (SMFE)," is presented in Chapter 8, while our proposal for analyzing relations, "Multiple Messages Relation Analysis (MMRA)," is in Chapter 9. Chapter 10 describes evaluations of our proposals in computer experiments, while Chapter 11 shows examples of anomaly detection using our methods. Chapter 12 presents a general discussion of our methods. Finally, we summarize our conclusion in Chapter 13.

# Chapter 2

# Cyberattacks against Vehicles

## 2.1 History of attacks against vehicles

Recent cars are electronically controlled by in-vehicle networks while they are running; electronic controls are also used to open and close their windows, etc. Furthermore, the number of cars connected to external networks such as the Internet are increasing. Such vehicles are called connected cars. An external network connection is needed for self-driving cars to download dynamic/static map information, traffic information, security patches, etc. Cars also have to be able to upload information.

However, an attacker can launch remote control attacks when a car is both electronically controlled and connected to an external network.

Remote control attacks are topics of active research. In 2010, Koscher et al.

showed that when attackers can connect their PCs to an in-vehicle network directly, they can control many functions of the car [15]. In 2011, Checkoway et al. showed that cars have many attack surfaces through which to launch remote control attacks [16]. In 2013, Valasek et al. showed that attackers could conduct remote control attacks against particular vehicles [17]. In 2015, Miller et al. demonstrated an attack that controlled an actual vehicle remotely [1]. In this attack, they hijacked an infotainment unit in a vehicle and rewrote the firmware of the unit. After that, they injected remote control messages into the hijacked unit. This attack led to a recall of 1.4 million cars, the first recall of cars for security reasons. In this dissertation, we consider methods of protecting against such attacks.

## 2.2 Types of cyberattacks against vehicles

Cyberattacks against in-vehicle networks can be classified into two varieties.

### 2.2.1 Direct attacks

This sort of attack uses a device that connects to the in-vehicle network. The attackers connect the attack device directly to a CAN entry point, such as the OBD-II port. Modified hardware from a general CAN controller may be used in this attack. In this case, the attack might only be detected by using the electrical

signal level. If the attacker can enter the car and connect the attack device to the in-vehicle network's entry point, this attack is a serious threat because the attacker can send arbitrary messages to the in-vehicle network directly. However, it is generally difficult to enter a car without a physical key. Taking countermeasures to prevent intrusions into the car (locking doors, etc.) makes such attacks difficult.

### 2.2.2 Indirect attacks

This sort of attack abuses the vulnerabilities of the electronic control unit (ECU) with a communication interface that connects to the Internet, etc. The attackers hijack the connection-ECU that connects to the external network by exploiting vulnerabilities in it and inject attack messages into the in-vehicle network from the external network through it. Although such attacks are limited compared with direct attacks because the ECU cannot be modified at the hardware level, the attacker need not enter the vehicle and can attack it remotely. To prevent such attacks, it is important to develop ECUs without vulnerabilities. However, as is the case with personal computers or smartphones, latent vulnerabilities can be discovered several years after the manufacturer has launched the product. Thus, this attack is a serious one because it is difficult to eliminate all vulnerabilities in ECU. Moreover, because the attack comes from an external network, and it hijacks an ECU (it includes the CAN controller) which then transmits the attack messages to

the in-vehicle network via the hijacked ECU, the attackers don't need to remodel any device in order to conduct indirect attacks.

Other types of indirect attack are introduced in [18], [19]. Attack methods against a passive keyless entry system (PKES) are presented in [18]. In these attacks, attackers relay control messages between the target car and its smart key by using relay devices (antennas, cables, and an (optional) amplifier). A successful attack allows them to open the door and start the car. In [19], a vulnerability in a smartphone application was abused. Because the application didn't authenticate the smartphone user, this attack can be realized by inputting part of Vehicle Identification Number (VIN) of the target car. In a demonstration, an attacker in Australia input the VIN of a target car in England, and took control of the target car remotely. There are many types of indirect attack that do not hijack any in-vehicle units.

## 2.3 Attacks dealt with in this dissertation

As mentioned above, this dissertation focuses on indirect attacks instead of direct ones where the attacker must physically enter a target vehicle. Direct attacks can be prevented by physically securing cars, for example, by locking their doors. In an indirect attack, the attacker need not to enter a target vehicle directly and can attempt to control the target vehicle remotely. Therefore, we think that indirect

attacks may have a larger impact than direct ones.

As explained in 2.2.2, there are many types of indirect attack. Among them, we will focus on ones that involve hijacking in-vehicle units, because this type of attack poses a threat to human life.

Many strategies against indirect attacks have been discussed in various conferences, papers, and standardization activities. These strategies are described below.

# Chapter 3

# Existing Security Strategies

Remote control attacks (indirect attacks) are serious threats to human life, and various security measures should be taken to protect vehicles against them. Security standards for general IT environments are described in [20] and [21]. This section describes the existing security strategies for general IT environments. Some of the strategies explained in this Section are described in [22].

## 3.1 Multi-layered security

The multi-layered security is a security strategy in which multiple techniques are implemented in layers for protecting an asset. The multi-layered strategy is introduced in [6][7]. Typical layers are explained as follows.

- Perimeter Security

This technique is used to protect against intrusions from external networks at the interface device. A typical example is a firewall.

- Authentication

  This technique is used to guarantee the legitimacy of the communication target. In many cases, cryptography, especially public-key cryptography, is used for authentication.

  Secure cryptographic algorithms must be used for secure authentication. The algorithms in Japan are recommended in [23]. Moreover, cryptographic modules must work correctly. That is, validation of cryptographic modules is important. The validation programs for cryptographic modules are described in [24] (for the U.S.) and [25] (for Japan).

- Detection Technology

  These technologies are used to detect attacks. A detection system embodying them is called an intrusion detection system (IDS). A detection system with a function that blocks network traffics is called an intrusion prevention system (IPS).

- Secure Boot

  This technique is used in boot phase of ECUs. It guarantees the legitimacy of stored executable codes. Cryptography, especially hash functions, is used

to achieve this guarantee. Secure boot technology is introduced in [26], [27].

- Secure Coding

  This is a implementation technique used in the development phase of security products. It enables the product to secure itself. Secure coding is introduced in [28].

- Security Update

  This technique is used to update the software and/or firmware of security products. Keeping security up to date is a very important consideration for current security products. In many cases, security patches are distributed from a central server. In this technique, cryptography is used for authenticating products and the central server. A typical example is a Windows update [29].

## 3.2 Defense in depth

Defense in depth is a security strategy for defending one asset. Multiple technologies, for example, monitoring the asset, authenticating user activities, and forensic recovery may be used. The difference between multi-layered security and defense in depth is described in [30]. A recommendation on defense in depth for an in-

dustrial control system is described in [8].

## 3.3 Security management cycle

In addition to applying security technologies like multi-layered security and defense in depth, a number of security management strategies have been proposed. A typical management strategy has been proposed by National Institute of Standards and Technology (NIST) [9].

The cybersecurity framework proposed in [9] consists of five phases, as follows (see also Figure 3.1). (Note that each phase is explained in [31].)

1. Identity

   This phase is for developing an organizational understanding to manage cybersecurity risks to systems, people, assets, data, and capabilities. This phase includes planning for asset management, the business environment, governance, risk assessment, and risk management strategy.

2. Protection

   This phase is for developing and implementing appropriate safeguards to ensure delivery of critical services. It includes identity management and access control, awareness and training, data security, information protection processes and procedures, maintenance, and protective technology.

3. Detection

   This phase is for developing and implementing appropriate activities to identify the occurrence of a cybersecurity event. This phase includes investigations of anomalies and events, continuous security monitoring, and detection processes.

4. Respond

   This phase is for developing and implementing appropriate activities to take actions regarding a detected cybersecurity incident. This phase includes response planning, communications, analysis, mitigation, and improvements.

5. Recovery

   This phase is for developing and implementing appropriate activities to maintain plans for resilience and to restore any capabilities or services that were impaired due to a cybersecurity incident. This phase includes recovery planning, improvements, and communications.

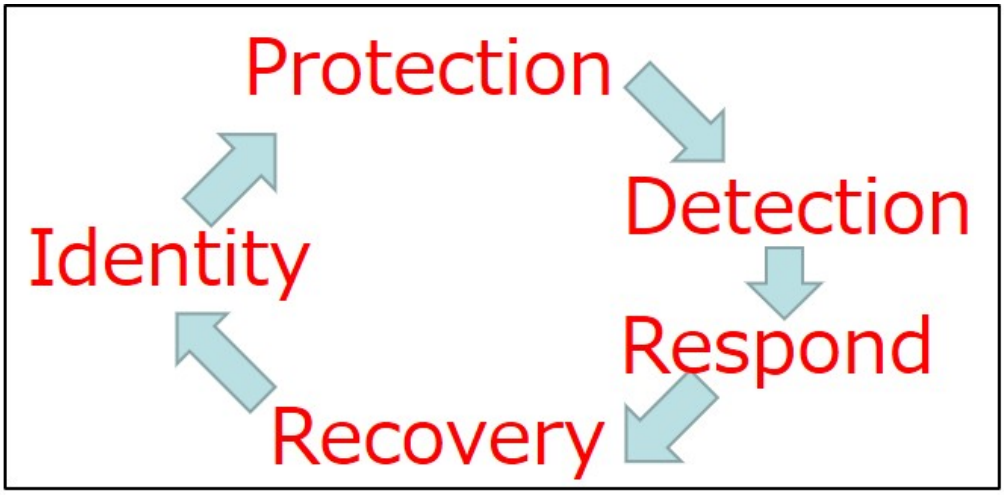


Figure 3.1: NIST Cyber Security Framework

# Chapter 4

# Strategies for Securing Vehicles

## 4.1 Standardization for connected car security

The strategies introduced in Chapter 3 are for general IT environments. Some techniques in those strategies may not be applicable to cars, because the protocols for in-vehicle networks are quite different from those of general IT environments. Moreover, damage from cyberattacks to general IT environments tends to be monetary, while cyberattacks on vehicles can lead to personal injury. Therefore, strong protection strategies and techniques are needed.

Strategies for securing vehicles have been discussed in many standardization organizations.

The UNECE World Forum for Harmonization of Vehicle Regulations (WP.29) of the United Nations Economic Commission for Europe (UNECE) has discussed

standardization of vehicle regulations against cyberattacks [2]. To meet these regulations, a cyber security management system and software update management system are needed. Approval authorities shall grant type approval to only vehicle that meet these regulations in near future.

Society of Automotive Engineers (SAE) is standardizing ISO/SAE 21434 that regulates vehicle life-cycle management [4]. Satisfying this standard is needed to meet the regulations of WP.29.

The regulations of functional safety ISO 26262 [5] state that supporting cybersecurity is mandatory in this standard.

Other standardization organizations have also devised cybersecurity regulations for vehicles.

The E-safety vehicle intrusion protected applications (EVITA) project regulates three hardware deployment architectures, called EVITA-full, EVITA-medium, and EVITA-light. [32] The difference between these architectures is in their security level. These regulations utilize hardware security modules (HSMs) to protect vehicles. The HSMs satisfy the regulations on security hardware developed by trusted computing groups (TCG). A typical TCG regulation is Trusted Platform Module (TPM) [33].

Automotive open system architecture (AUTOSAR) [34] is a worldwide development partnership of vehicle manufacturers, suppliers, service providers and companies. AUTOSAR standardizes in-vehicle software regulations including se-

curity technologies like Message Authentication Codes (MAC).

Japan Automotive Software Platform and Architecture (JasPar) is an organization that discusses and standardizes security technologies [35]. In particular, JasPar discusses detection technologies, over-the-air update, vulnerability testing, etc.

Cybersecurity for vehicles is researched in the following articles.

[36] is a research report on automotive security from the Japan Automobile Research Institute (JARI). [37] describes the activities of the Information-Technology Promotion Agency, Japan (IPA), and IPA has published a guide to vehicle information security [38].

Although the above standardization activities have emphasized the ways in which cybersecurity for vehicles is important, they have standardized the strategies not any individual technique.

## 4.2 Multi-layered security for vehicles

The multi-layered security introduced in Section 3.1 is for a general IT environment. Here, we discuss multi-layered security for vehicles. An example of a multi-layered security strategy for vehicles is described in [39]. Typical techniques of multi-layered vehicle security are as follows (see Figure 4.1).

- Perimeter Security

This kind of security is similar to that for a general IT environment. An example is a firewall.

- Authentication

  Authentication is also mainly the same as in an IT environment. An example is authentication based on cryptographic techniques. The use of message authentication code (MAC) for vehicles is discussed in [34]. However, there are performance limitations to the protocol used in current in-vehicle networks. MAC cannot be used on all messages, only some of them. Therefore, various techniques for other layers must be combined in order to secure vehicles.

- Detection Technology

  This technique is used to detect attacks. Uses of IDS and IPS are discussed in [35]. In general, the detection techniques used in IT environments cannot be used in in-vehicle networks, because the transmission protocols are quite different (TCP/IP for general IT environment and controller area network (CAN) [40], local interconnect network (LIN), etc., for in-vehicle networks). Therefore, specialized detection techniques are needed for in-vehicle networks.

- Secure Boot

This technique is similar to the ones for IT environments, such as secure boot or trusted boot (using TPM).

- Secure Coding

  This is a technique in which ECU secures itself. It is like the one used in an IT environment.

- Over the Air Update (Security Update)

  This technique is used to update the firmware of ECUs. The patches are received via wireless communication from a central server. Issuing of security patching for vehicles via wireless communication is called over-the-air update (OTA). OTA technology is described in [41].

### 4.2.1 Application of IT technologies for in-vehicle environments

Some of the above techniques are the same as those for IT environments. Perimeter security can be prepared by applying technology similar to firewalls to in-vehicle networks. Since the in-vehicle equipment are devices in the IT environments, secure boot can be prepared. Since secure coding is a coding method, it can also be used in vehicle environments. Specialized solutions for vehicles are needed for detection, over-the-air update, and part of the authentication. We consider that performance limitations are why vehicular-specialized solutions of the

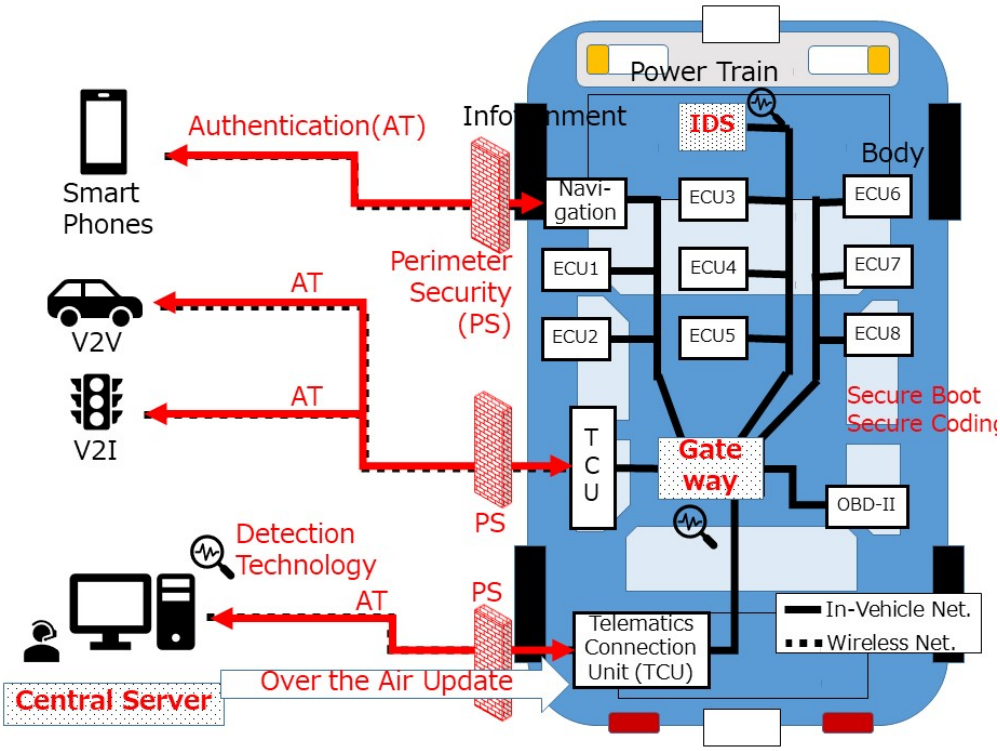


Figure 4.1: Multi-layered security for vehicles

over-the-air update and authentication are needed. These performance limitations are mitigated by improving the performance of the in-vehicle equipment. Therefore, this problems is replaced to the cost problem. However, differences between protocols is an essential reason why vehicular-specialized solution of the detection are needed. Because typical protocol used in in-vehicle network is quite different from the protocol used in IT environment (for example, there is no 'From' or 'To' field). This problem can be solved by development of vehicular-specialized technique. So, we think developing detection technologies are the most important

among them. The detections may be performed at the gateway, by a specialized detection unit, and/or the central server. Here, we will focus on detection of remote control attacks (indirect attacks) on in-vehicle networks especially CAN networks.

## 4.3 Security management cycle for vehicles

In this section, we discuss detection technology by using a security management cycle like in [9]. Here, he security management cycle for in-vehicle networks is similar to the one in [9], but consists of the four phases shown in Figure 4.2. The four phases are explained as follows.

1. Protection

   The protection methods include multi-layered security techniques described in Section 4.2, perimeter security, authentication, secure boot, and secure coding.

2. Detection

   When the protection methods are not able to protect the vehicle because of some vulnerability, attacks are detected by using specialized in-vehicle technology.

3. Respond

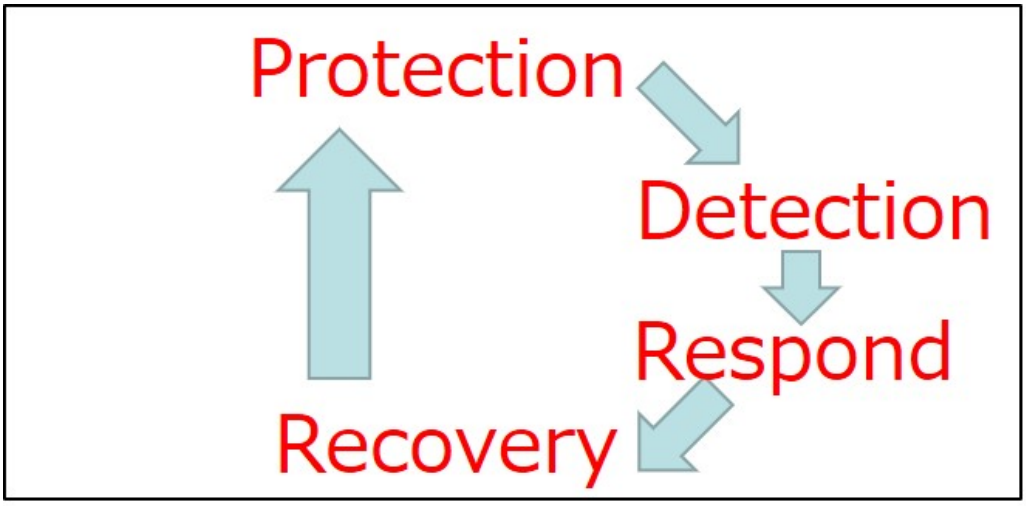


Figure 4.2: Security management cycle: attack detection

In order to recover from a detected attack and restore the correct activity of the vehicle, the vehicle may gradually stop at the side of the road. The functions for this phase should be developed by car manufacturers.

4. Recovery

   When the vehicle restores its activity, log data on the attack are sent to the central server. There, the log data are analyzed in order to develop security patches. The developed patches are then transmitted to the vehicle by using OTA update.

The above cycle includes the operations of all vehicles and the central server. We assumed that the identification step is considered in the development phase of vehicles. Moreover, it is considered by operators of the central server after the

operation is started. We think that this cycle indicates automatic operations, so an identification step is omitted in the figure.

Among them, we think that protection and recovery can be handled by implementing the conventional IT security techniques introduced in Chapter 3. The respond can be realized by technology developed for vehicles by the manufacturers. On the other hand, we think that detection requires specialized in-vehicle techniques.

# Chapter 5

# In-vehicle Networks

## 5.1 Varieties of in-vehicle network

There are various protocols for in-vehicle networks. Typical protocols are described in Table 5.1. The Controller Area Network (CAN) [40], CAN with Flexible Data Rate (CAN-FD) (An extension of the original CAN), FlexRay [42], and Ethernet (100 base T1) [43] are prepared for vehicle controls. However, CAN-FD, FlexRay, and Ethernet are intended for future vehicles. Current vehicles can only use CAN. In the automotive industry, new technologies are deployed only after long-term evaluations. Therefore, we think that CAN will be used for a while to come. Thus, we will focus our discussion on detecting attacks on in-vehicle networks using CAN.

Table 5.1: Typical protocols for in-vehicle networks

| Name | Main Purpose | Maximum Speed |
|---|---|---|
| Controller Area Network (CAN) [40] | vehicle control | 1 Mbps |
| CAN with Flexible Data Rate (CAN-FD) [40] | vehicle control | 8 Mbps |
| Local Interconnect Network (LIN) [44] | body control | 20 kbps |
| Media Oriented Systems Transport (MOST) [45] | multimedia | 150 Mbps |
| FlexRay [42] | vehicle control | 10 Mbps |
| Ethernet for Vehicles (100 base T1) [43] | vehicle control | 100 Mbps |

## 5.2 Controller area network (CAN)

### 5.2.1 Overview of CAN

CAN is a network protocol used in control systems. It was developed by Bosch and is standardized as ISO11898 [40]. CAN can be applied to many vehicles. Network nodes called Electronic Control Units (ECUs) that use CAN communicate with each other by using the voltage difference between two communication lines. Because the voltage differential does not change much, CAN is resistant to electronic noise on the lines. In-vehicle networks generally have a bus topology. When an ECU transmits a message to the in-vehicle CAN bus, the message reaches all ECUs connected to that bus. Moreover, when ECU (a) tries to trans-

mit a message to the in-vehicle network while a message is being transmitted by another ECU (b), ECU(a) retries its transmission of the message after ECU (b) completes its transmission. In CAN, bit ‘0‘ is called dominant and ‘1‘ is called recessive. The dominant bit has priority over the recessive bit. For example, in case of a message is sent from another ECU (b) at the same timing, the transmitting bit becomes “dominant”, and ECUs whose transmitting bit is recessive pend their transmissions. This mechanism is called Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). CAN has four data format types (data frame, remote frame, error frame, and overload frame). The data frame is used for general data transmissions. This dissertation focuses on the data frame.

### 5.2.2 Data frame

The data frame contains a standard format and extended format. The standard format has an 11-bit ID field, and the extended format has a 29-bit ID field. Although we will only deal with the standard format, our discussion is also applicable to the extended format. The data frame is explained in Figure 5.1. We call the value of the ID field the “CAN-ID”. Each CAN-ID is determined in the car development phase for each vehicle type. CAN-ID indicates the meaning of the message and is used for communication arbitration. Communication arbitration is a mechanism whereby only the high priority message is processed when two or more ECUs

transmit data frames at the same time. Because the dominant bit is given priority over the recessive bit, the message where the dominant bit appears first in the most significant bit (MSB) of the CAN-ID is processed first in all transmitted frames. When the dominant bit appears in the MSBs of two or more CAN-IDs simultaneously, priority is similarly determined by the subsequent bit. The transmission node transmits messages to and simultaneously observes messages from the CAN bus. It stops transmitting if the CAN-ID of the message sent from another ECU at the same timing has higher priority than that of the message it is transmitting. In general, CAN-IDs are managed such that messages sent from different transmitting ECUs always have different CAN-IDs. In addition, each ECU receives only designated messages that are determined on the basis of the CAN-ID. Other received messages are ignored. The data frame also has a data field for storing data. For example, CAN-ID 0x100 means an engine message whose data field may contain the angle of the accelerator pedal. The data field can have 64-bit data at maximum. The length of the data is specified for each CAN-ID. The length is stored in the data length code (DLC) field of the CAN message. Here, we will focus on data frames, because attackers send data frames in order to remotely control vehicles. We call the data field the "payload". A payload can be divided into sub fields, and each sub field stores one piece of data. We will sometimes call this data "field data". For example, the top field in the data field consists of 8 bits and its field data is constant, while the subsequent field consists of 8 bits and its

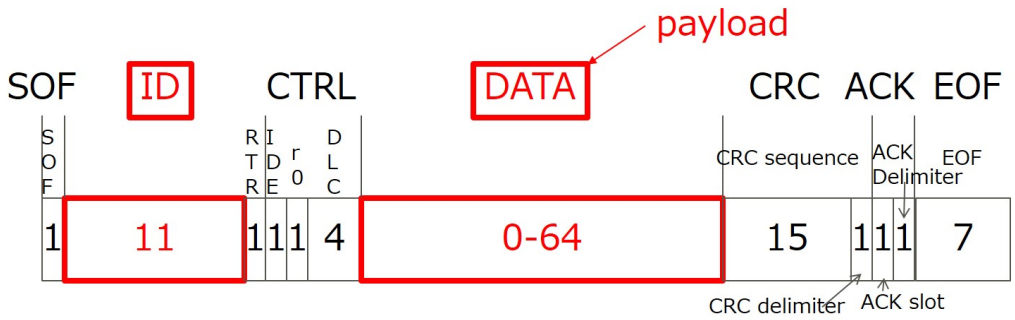


Figure 5.1: CAN data frame

field data may change little by little. The bottom field consists of 8 bits and its field data is a checksum value. The boundaries between sub fields will be called boundaries. Each type of field data has a behavior, such as “constant”. We thus call the behavior of each type of field data a “Type”. The position and length of the field data is represented by a tuple $(position, length)$. The stored pattern of a payload is called “format”. In general, a format is determined for each CAN-ID. These parameters are illustrated in Figure 5.2.

### 5.2.3 CAN message periodicity

Many messages are transmitted at the designated transmission cycle determined for each CAN-ID. There are various cycle intervals, which range from about ten milliseconds to ten seconds. We call these messages periodic messages. Additionally, there are event-based messages and non-periodic messages. Event-based messages are transmitted periodically in a normal situation. However, when an

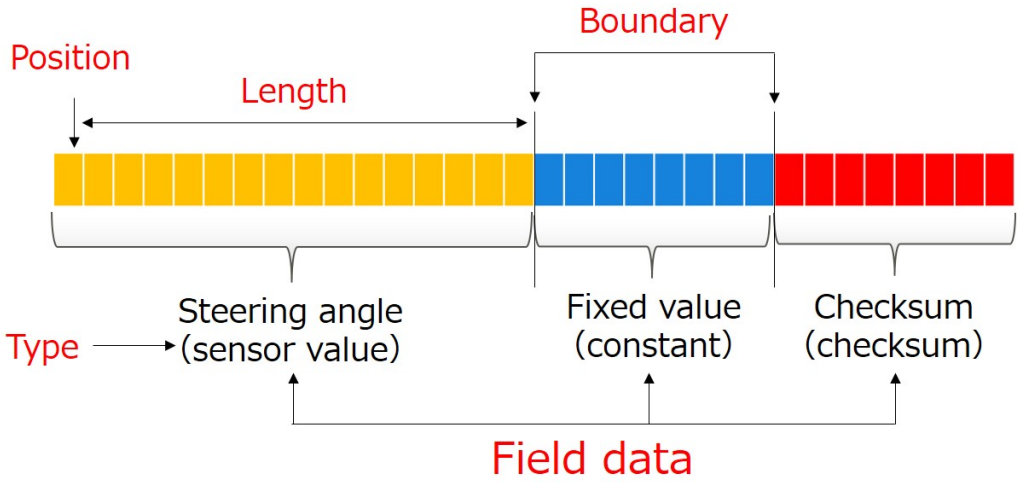


Figure 5.2: CAN format of the data field

event of some kind occurs, the messages are not transmitted periodically. Non-periodic messages, on the other hand, are always transmitted non-periodically. Each CAN message corresponds to one of these four patterns. The patterns for each CAN ID are shown in Figure 5.3. Note that periodic messages account for more than 60% of all the messages.

- Periodic messages:
  - Perfectly periodic messages: The reception interval of perfectly periodic messages may deviate only slightly. Namely, the reception intervals recover to normal status within a fraction of cycle. The number of messages received for each CAN-ID is constant during a short observation period; i.e., there is no shortage or excess of messages.

  - Quasi-periodic messages: In this case, the message reception intervals may temporarily have large deviations. One cause of temporary deviation is message arbitration. The reception intervals recover to normal status after some cycles. The number of messages received for each ID is constant during a long observation; i.e., there is no shortage or excess of messages.

- Event-based messages: Messages are sent to the CAN bus periodically in normal situations. However, when events occur in the ECU, event-driven messages are sent to the CAN bus. From then onwards, messages are sent periodically again to the CAN bus.

- Non-periodic messages: In this case, the message reception intervals show large deviations to the extent that intervals cannot be determined at all. The number of messages received for each CAN-ID is not constant over a long observation period.

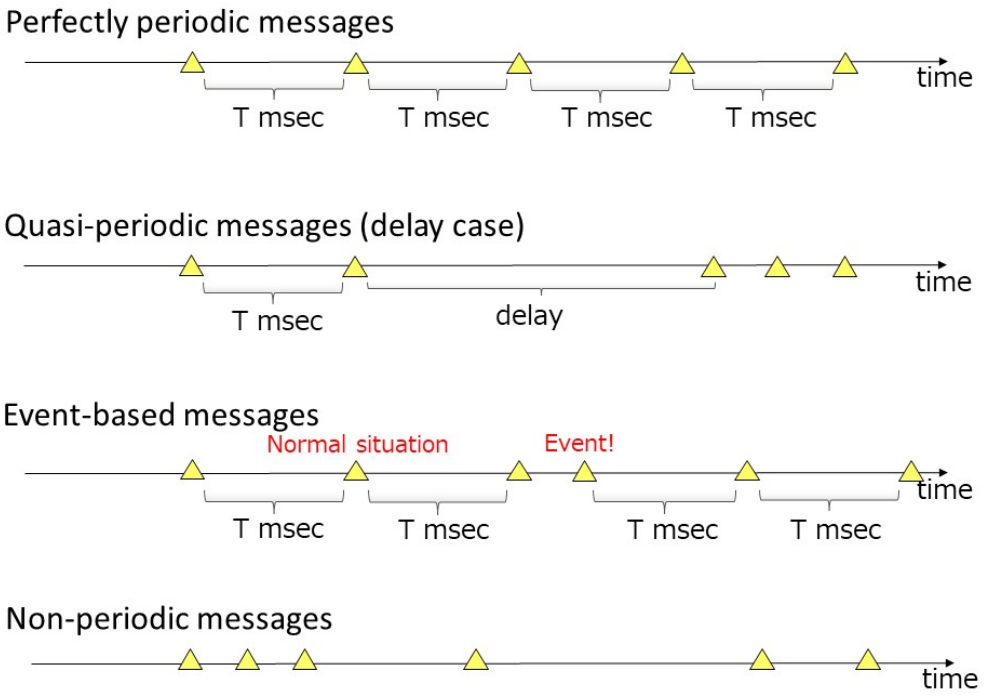


Figure 5.3: Message patterns categorized by transmission period

# Chapter 6

# Detecting Attacks on CAN

## 6.1 Abstract of CAN anomaly detection

This Chapter discusses the detection of attacks on a CAN network, which is understood as anomaly detection in in-vehicle security.

We will deal with two anomaly detection strategies, namely, rules-based and machine-learning-based. Note that other techniques not described below are reported in [46], [47].

### 6.1.1 Rule-based detections

In rules-based detection, the detection rules are pre-determined. To construct the rules, many CAN logs should be collected beforehand and their features analyzed. Detection is performed in accordance with the constructed rules. The framework

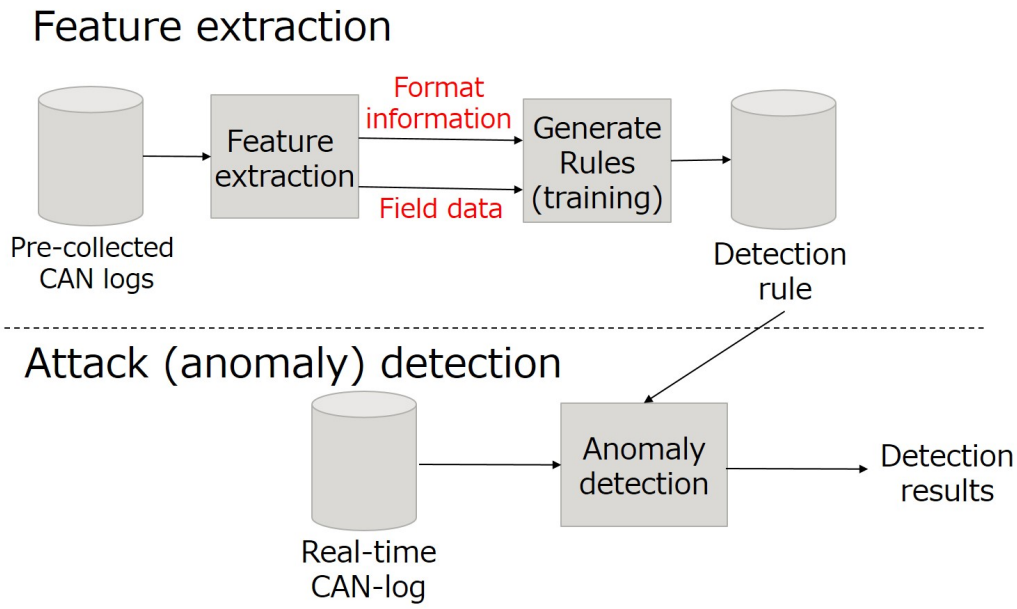


Figure 6.1: Framework of CAN anomaly detection

of this strategy is illustrated in Figure 6.1. There are two phases: feature extraction and anomaly detection. In the feature extraction phase, the features of the CAN messages are extracted from the collected CAN logs. These features may include the formats of the CAN messages. After that, detection rules are constructed from extracted features. In the anomaly detection phase, CAN messages collected in real-time are judged as normal or anomalous in accordance with the constructed detection rules. The following are approaches that can be used to extract the features.

**Focusing on message reception periodicity**

This approach can be applied to periodic messages; when the reception intervals differ from the normal values, the situation is judged to be anomalous. Examples of using this approach are shown in [10], [11], [48]. This approach is considered to be effective on CAN messages because they are predominantly periodic.

**Focusing on the behavior of field data in the CAN payload**

This approach can be applied to messages of all categories, including event-based messages and non-periodic messages. There are two phases, a feature extraction phase and a detection phase. In the feature extraction phase, the normal behavior of each piece of field data in the CAN payload is estimated. After that, detection rules are constructed by utilizing the estimated behavior. In the detection phase, when a received message matches one of the constructed rules, the situation is judged as anomalous. An example of using this approach is shown in [49]. The approach is effective for all messages include periodic, event-based, and non-periodic messages.

**Focusing on event based messages**

This method analyzes relations between data and can be applied to event-based messages. To the best of our knowledge, no one else has proposed an approach exactly like this one even though some methods aimed at event messages have

been presented. An example of the detection method against event messages is shown in [50]. In that method, the detector utilizes correlations of the field data and their periodicity. However, their event messages differ from our definition of them. In our definition, the event messages are non-periodic.

### 6.1.2 Non-rule-based detection

In non-rule-based detection, no specific detection rules are determined. Detection is performed in accordance with prediction models. A typical detection method of this strategy is machine-learning-based detection.

#### Machine-learning-based detection

This approach can be applied to all messages. Here, detectors train a machine-learning model using many observed messages (this is called the training phase). The detection accuracy of the trained model is confirmed by running it on test messages (the test phase). The tested model is then used to check the received messages (the prediction phase). Examples of using this approach are shown in [51], [52], [53]. There are two main kinds of training method, supervised learning and unsupervised learning. Unsupervised learning is suitable in many cases of anomaly detection. The framework of this strategy is also illustrated in Figure 6.1. In the feature extraction phase, the features of the CAN messages are extracted from the collected CAN logs. These features may include the formats of the CAN

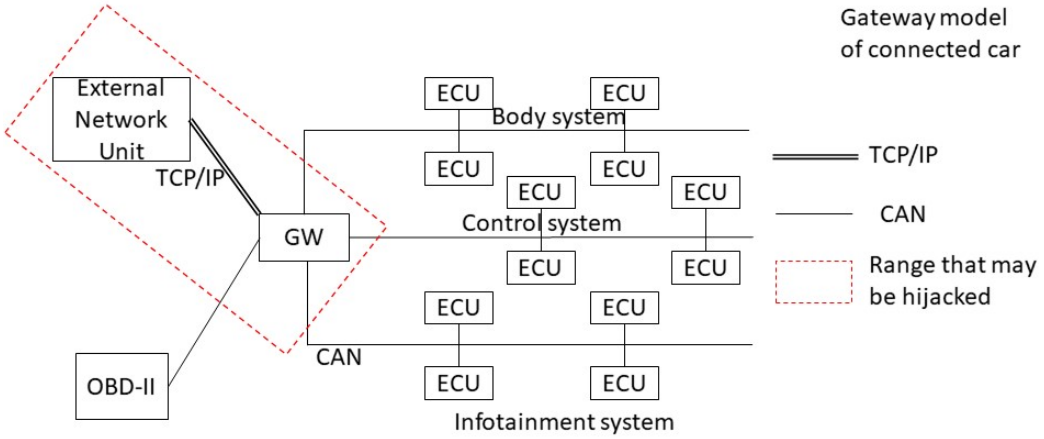


Figure 6.2: Typical network without cybersecurity features

messages. After that, the extracted features are used to train the machine-learning model. The detection is conducted with the trained model.

## 6.2 Network construction

In this section, we discuss the construction of an in-vehicle network. Modern in-vehicle networks of connected cars follow the "gateway model". An example of constructing a gateway model is introduced in [36]. Figure 6.2 shows an example of a network following the gateway model. In this model, the attack surface of the network is the external network unit. In many cases, the gateway is connected to an external network unit by TCP/IP, so attackers may be able to hijack the gateway via a TCP/IP network. However, attackers cannot hijack any ECUs via

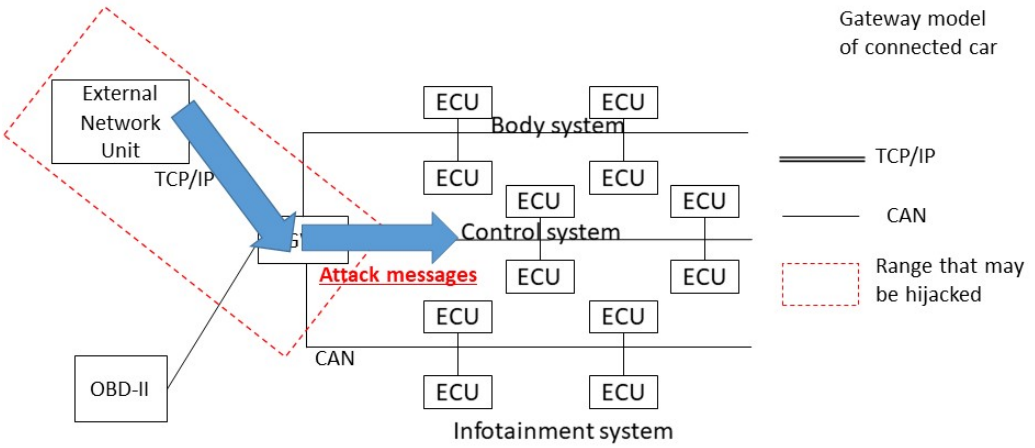


Figure 6.3: Indirect attack on a network

a CAN network, because usually only sensor data are transmitted in the CAN protocol. That is, no instruction codes are transmitted via CAN. Moreover, it is very difficult to hijack ECUs by changing the sensor values in the CAN payload. Therefore, we will exclude the threat of hijacking ECUs via a CAN network from our considerations.

In order to remotely control the target vehicle, we think that attackers would likely inject attack messages into the CAN network via a hijacked gateway. This attack pattern is described in Figure 6.3. Here, attackers inject attack messages via a hijacked external network unit and gateway.

This attack can be prevented by setting an intrusion prevention system (IPS) between the gateway and control system of the CAN network. This setup is illustrated in Figure 6.4. The IPS detects attacks by checking the CAN-ID. When

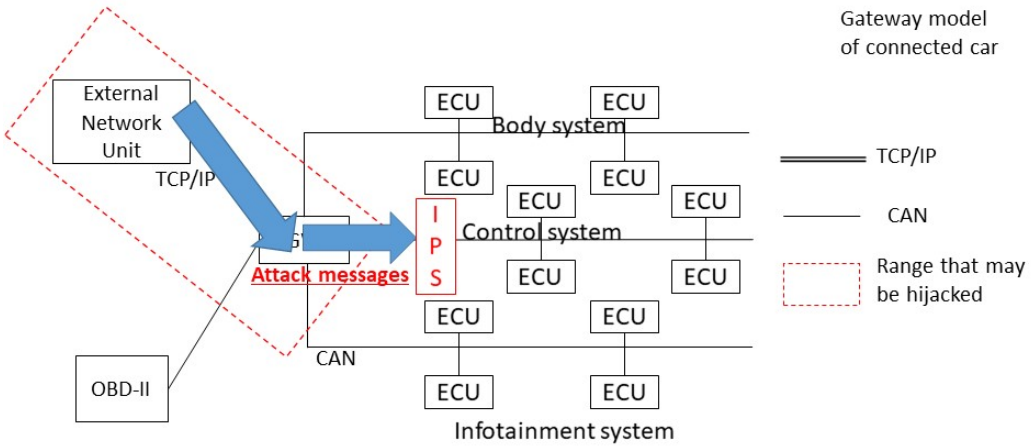


Figure 6.4: Construction of network with an intrusion prevention system (IPS)

messages with a CAN-ID that is not used in the control system are received, the IPS prevents those messages from passing through it. On the other hand, when messages with a CAN-ID that is used in the control system are received, the IPS checks the reception periodicity of the messages with the same CAN-ID or the content of the messages. It does so because an attack may disturb the periodicity of legitimate messages that are perfectly or quasi-periodic, as described in Section 5.2.3. Therefore, periodicity should be checked in these cases. On the other hand, when an attack message with a CAN-ID that is the same as an event-based or non-periodic message is injected, checking the periodicity will not be effective, and instead, the content of the messages should be checked. In many cases, because the IPS delays messages passing through it, an intrusion detection system (IDS) should be used instead of the IPS.

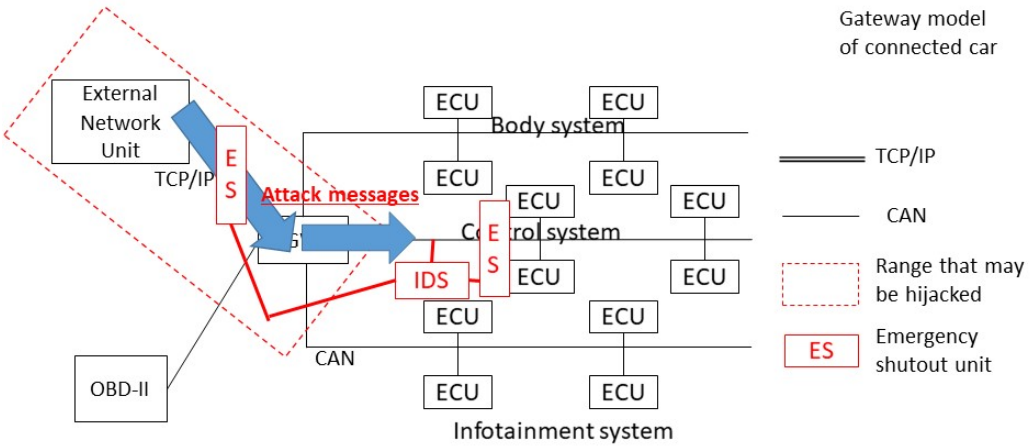


Figure 6.5: Construction of network with an intrusion detection system (IDS) and emergency shutout system (ES)

A network constructed using the IDS is illustrated in Figure 6.5. The IDS cannot prevent messages from passing through it. Therefore, we think an emergency shutout unit (ES) should be used in combination with the IDS to block messages when the IDS detects an attack.

The above discussion is for connected cars. Now, we discuss the construction of a network specifically for self-driving cars. A network for self-driving cars is illustrated in Figure 6.6. This figure is illustrated by our inference. The manual driving system and self-driving system are separated by a switch.

To prioritize manual driving, messages with high-priority CAN-IDs may be assigned to the manual driving system. In this case, an attack with the CAN-IDs used in a self-driving system is not a threat because the attack is disabled when

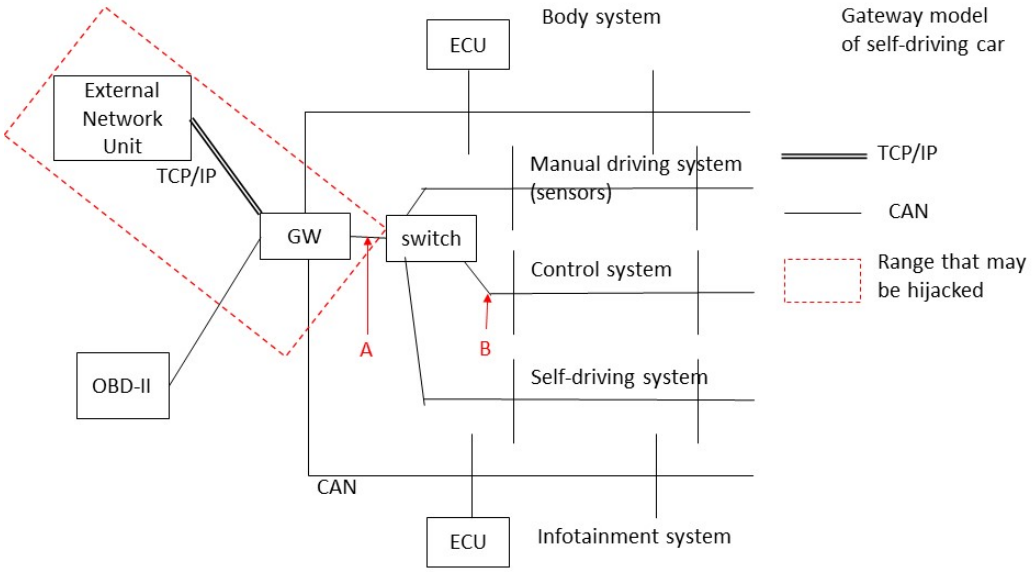


Figure 6.6: Network for a self-driving vehicle

the driver controls the vehicle by using a manually controlled pedal or handle. On the other hand, an attack with a CAN-ID for a manual driving system poses the same threats as discussed above.

Another case is one in which the same CAN-IDs are assigned to the manual driving and self-driving systems. Here, switch control messages pass through the switch. An IPS or an IDS with an ES setting between point A and B in Figure 6.6 can prevent attacks in both situations discussed above.

The final case, illustrated in Figure 6.7 includes the gateway and a self-driving unit connected to a TCP/IP network for receiving various information from it. In this case, the self-driving unit may be hijacked via the TCP/IP network. When an

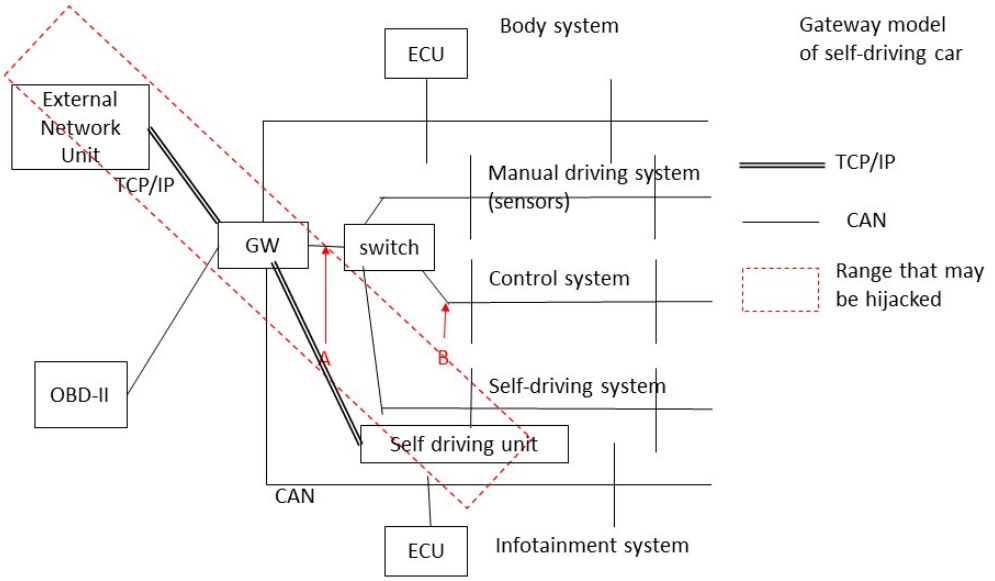


Figure 6.7: Another network for a self-driving vehicle

attack with messages of the same periodicity or with messages without contradictory content is launched against the gateway and self-driving unit, the IPS (or the IDS with the ES) set between points A and B cannot detect the attack. In this case, the secure boot technology should be used to prevent tampering the self-driving unit. Otherwise, the vehicle must make an emergency stop, which may require an emergency stop unit to be installed. Now, let us describe the requirements of these detection technologies.

## 6.3 Requirements of attack detection technology

The discussion in Section 6.2 shows that detection technologies are needed for all types of messages described in Section 5.2.3, namely, perfectly periodic, quasi-periodic, event-based, and non-periodic.

The following are typical strategies for detecting attacks.

- Checking for contradictory messaging behavior

  This strategy can be applied to all types of messages that have content. Because it focuses on content of the message, this strategy needs information on the expected content of the message to judge whether a message is anomalous. This information includes the area in the CAN message where the content is saved, the ranges of the field data, and the reception periods of the messages. The effectiveness of the detection method depends on the knowledge of them. The existing detection techniques using this strategy are described in [50], [54].

- Checking for contradictory message reception periodicity

  This strategy focuses on the periodicity of the received messages. Accordingly, it is applicable to perfectly periodic messages. Moreover, using it in combination with the strategy of analyzing messaging behavior is better than using only messaging behavior. The detection techniques using this

strategy are described in [10], [11].

We consider that detection technologies for perfectly periodic and quasi-periodic messages are very important for the following reasons:

- These messages occupy over 60% of all messages;
- Many of the control messages are sent periodically.

Methods that check inconsistently of reception periodicities are useful for these messages. However, the detection accuracy of the existing methods described the above is not so high. Our detection method, hereafter called method 1 or proposal 1, checks the periodicity of the received messages and is more accurate that the existing methods.

As well, we cannot ignore the event-based and non-periodic messages because they nonetheless occupy a significant percentage of the total number of messages. By using our proposals 2 and 3, we can construct some detection techniques for these messages.

In particular, Proposal 2 analyzes the behavior of each message. Although some previous methods can do this, our proposal can identify behaviors with higher accuracy. Proposal 3 finds relations between field data in multiple messages, something which previous research has not treated.

The relation of our methods are described in Figure 6.8. The width of each column of target messages indicates the percentage of existence of each message.

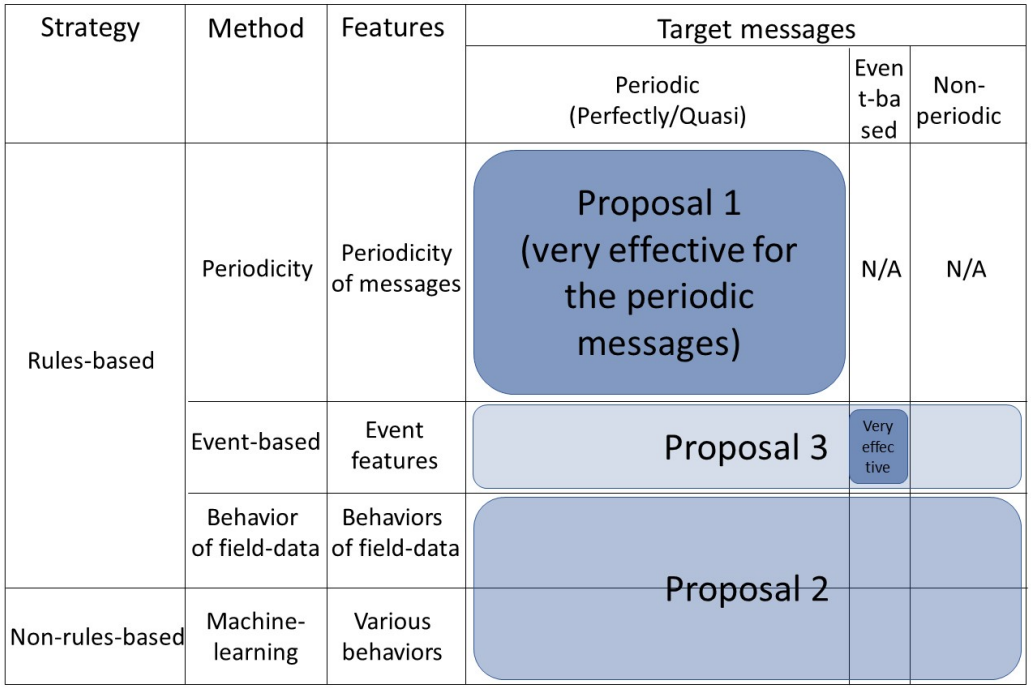


Figure 6.8: Relation of our methods

The density of the highlight indicates the effectiveness of each method. For the periodic messages, Proposal 1 is the most effective among our methods. However, this method cannot be applied to the event-based and the non-periodic messages. For the event-based messages, detection methods using Proposal 3 is the most effective. Proposal 2 and 3 can be applied to all messages include the non-periodic messages. However, we think that detection methods using Proposal 2 can detect attacks with higher accuracy for the periodic and non-periodic messages compared with that using Proposal 3.

## 6.4 Our detection methods

### 6.4.1 Abstract of our proposal

We propose the following methods.

- Proposal 1

  A new rule-based anomaly detection method, called “Anomaly Detection by Cumulative Sum”, for periodic (perfectly periodic and quasi-periodic) messages.

- Proposal 2

  A new format estimation method, called “Single Message Format Estimation,” that is useful for constructing rules-based and machine-learning-based detection methods for all types of messages (perfectly periodic, quasi-periodic, event-based, and non-periodic messages).

- Proposal 3

  A method of analyzing relations between field data in multiple messages, called “Multiple Messages Relation Analysis,” that is useful for constructing rules-based detection methods for all types of messages (perfectly periodic, quasi-periodic, event-based, and non-periodic messages).

By using the above methods, detectors can construct detection methods for all types of messages.

The anomaly detection by cumulative sum (proposal 1) can detect attacks on perfectly periodic and quasi-periodic messages, with almost no false positives and negatives. The anomaly detection by cumulative sum method is described in Chapter 7. The single message format estimation (proposal 2) and multiple messages relation analysis (proposal 3) work on all types of messages. The single message format estimation is described in Chapter 8, while the multiple messages relation analysis is described in Chapter 9. These methods can be used to build both rules-based detectors and machine-learning-based detector.

All of our methods can be applied to periodic messages they are our main targets. Multi-layer use of our methods enables detecting attacks with more high accuracy than single-layer use of each method.

# Chapter 7

# Proposal 1: Anomaly Detection by Cumulative Sum

In this chapter, we described a new rules-based anomaly detection method called "Anomaly Detection by Cumulative Sum". This method was proposed in the domestic conference [55] and the Embedded Security in Cars Conference (escar) Asia [56], [57], which is one of the three largest international conferences on vehicle security research. Our company has applied for patents of this technique [58], [59], [60]. A press-release about this method was also published [61]. The automotive industry is interested in this method because it has almost no false positives or negatives in the assumed situations. Here, we will focus on perfectly periodic and quasi-periodic messages that may be long delayed and early arriving messages. We will assume that attacks against event-based and non-periodic

messages are detected by using other techniques.

## 7.1 Target messages

As explained in Section 5.2, only the messages with the highest priority CAN-IDs flow to the CAN bus when two or more messages are transmitted from different ECUs at the same time. When a certain ECU transmits a message $M_1$ whose CAN-ID has higher priority, if another ECU has already started transmitting the ID field of message $M_2$, message $M_1$ is transmitted only after message $M_2$ even if CAN-ID of $M_2$ has lower priority. This causes a delay in message transmitted from the ECU to the in-vehicle network. Methodologies that forecast the maximum delay time due to such data collisions have been reported in [62], [63], [64], etc. Moreover, the targets of the existing detection methods explained in Section 7.2 are the perfectly periodic messages described in Section 5.2.3. By using the existing methods for short delayed and early arriving messages, false negatives can be minimized and attacks detected with high accuracy. However, in actual vehicles, long delays and early arrivals may sometimes occur in normal situations. In these cases, many of the existing methods cannot detect attacks correctly. We thus decided to construct a method that can detect attacks with extremely high accuracy with very few false negatives, even in situations with long delays and early arrivals. Namely, our method targets not only perfectly periodic messages,

but also quasi-periodic messages as described in Section 5.2.3.

## 7.2 Existing methods

This section explains three well-known detection methods for periodic messages in CAN. As explained in Section 7.1, most suffer from false positives or negatives in the case of significantly delayed and early arrivals.

### 7.2.1 Cycle detection

This sort of method observes only the interval between two consecutive messages. We hence call it cycle detection. A cycle detection method is described in [10]. The method determines whether the intervals between messages differ from those of normal communications. Specifically, the arrival time of the following message under ideal reception conditions is ascertained and the permissible boundary is set on the basis of it. The subsequent message is judged as a normal message if its reception time is within the permissible boundary and as an attack if it falls outside of the boundary. Cycle detection is illustrated in Figure 7.1. In this example, when the interval between the reception times of two consecutive messages exceeds the predetermined permissible boundary, the situation is judged to be an attack. The permissible boundary is updated by using the current reception time of the message when the time is within the permissible boundary.

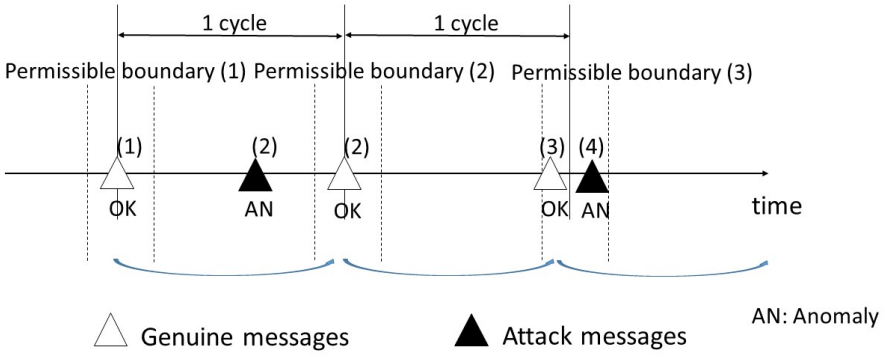


Figure 7.1: Example of cycle detection

### 7.2.2 Delayed-decision cycle detection

Delayed-decision cycle detection [11] is an algorithm proposed by Otsuka et al. This method observes the intervals of reception times for about three consecutive messages. Its algorithm considers the possibility of fluctuation of the message transmission cycle. This method is explained in Figure 7.2. Two parameters named $\alpha$ and $\beta$ are used. $\alpha$ is used to decide which message is to be a detection target. The reception time $t_1$ of a certain message is not detected as an attack when the time interval between the previous reception time $t_0$ and $t_1$ exceeds $T-\alpha$ ($T$ is the cycle of the message). When $t_1-t_0$ is less than $T-\alpha$, the detection judgement is reserved and reception of the next message is awaited. When the reception time of the next message $t_2-t_0$ is less than $T+\beta$, this situation is detected as an attack. As a result, this method can detect attacks that occur during both delays and early arrivals.

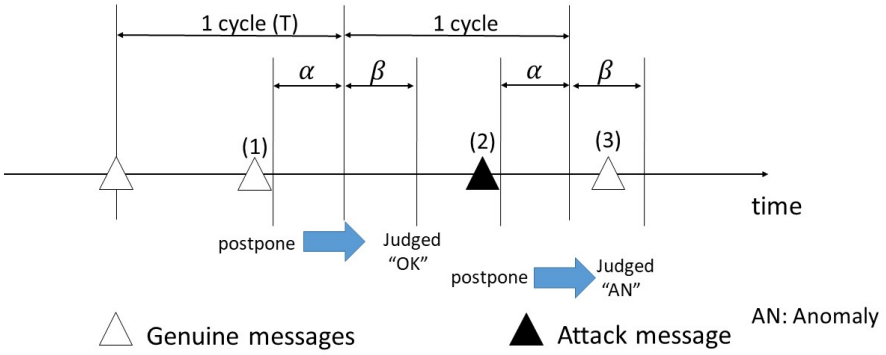


Figure 7.2: Delayed-decision cycle detection

### 7.2.3 Waszecki's method

In the method proposed by Waszecki et al. [48], a detector derives the worst-case jitter $j$, which means the maximum deviation from the designated reception cycle. After that, it derives the minimum time interval between consecutive messages $\delta$, burst capacity $\nu$, and decrement value of the counter $\rho$. If the detector can derive all of them, this method can detect attacks with high accuracy. The detector calculates the counter and timer. $\rho$ is subtracted from the counter when the message is received. The timer indicates the timing of the increments of the counter. If the counter is less than zero, the situation is judged as an attack. This method seems similar to our method at first glance. However, it differs from our method in the following ways.

- Need for a jitter calculation: In their method, all parameters must be set correctly. Many of them can be calculated using the formulas shown in [48].

Only jitter must be determined by the detector. Similarly, in our method, all parameters must be set correctly. However, in most cases, the parameters can be easily selected in our method.

- Treatment of counters after detection: Their paper does not describe the treatment of counters after an attack is detected. After an attack is detected, the counters are sure to have low values than during normal use. This may cause their method to give false positives or negatives when it continues to process without increasing the value of the counter.

## 7.3 Proposed method (anomaly detection by cumulative sum)

As mentioned above, we call our method anomaly detection by cumulative sum (ADCS). ADCS can detect attacks when the messages are perfectly periodic or quasi-periodic, as described in Section 5.2.3.

### 7.3.1 Assumptions

Anomaly detection by cumulative sum makes the following assumptions.

1. All attack messages are injected from external networks via external communication units or via the OBD-II port.

2. Attackers cannot hijack ECUs via in-vehicle networks with the CAN protocol.

3. The externally connected ECUs are not related to the essential systems of the car, such as the control system.

4. Messages may have some delays or early arrivals.

5. All messages are received.

These assumptions are reasonable, as follows. Regarding assumption 1, we assume indirect attacks. Regarding assumption 2, only sensor data (excluding operation codes) are sent on in-vehicle networks with CAN, so hijacking ECUs via CAN networks is almost impossible. Regarding assumption 3, in many vehicles, control ECUs send out all of the control messages. Therefore, attack messages from external networks are additionally transmitted with normal control messages. In other words, no message replacement attacks can be conducted. Regarding assumption 4, we assume that the messages are perfectly periodic or quasi-periodic. Regarding assumption 5, our experiments verified that no messages are lost. Detailed considerations are discussed in Section 12.1.1.

### 7.3.2 Basic idea

ADCS uses two counters. The first counter, $n$, indicates the number of received messages, and it is incremented when a message is received. The second counter, $x$, indicates the expected number of messages to be received; it is incremented at the end of each cycle. Timing of the end of each cycle is determined using the offset ratio $p$. The end of the first cycle is calculated by $p \times T + T + t_0$ such that $t$ is the reception time of the first message. After that, the end of each cycle is calculated by $t + p \times T + k \times T(k = 2, ...)$. When $p$ is set 0.5, the end of each cycle is just middle between two consecutive expected reception times. So, we will easily set $p = 0.5$. The basic idea is that a situation where $n > x$ is considered to be an attack. Figure 7.3 shows an example of attack detection when an attack is injected into a periodic transmission message. In this example, the attack is injected when the expected number of received messages is 12. The relationship between the two counters becomes $n > x$ and the attack is detected. After the attack is detected, the counter $n$ is decremented.

### 7.3.3 Delayed messages

The basic idea presented in Section 7.3.2 can be used when genuine messages are delayed. Figure 7.4 shows an example of ADCS where genuine messages are delayed. In this example, two messages are delayed when the expected numbers of

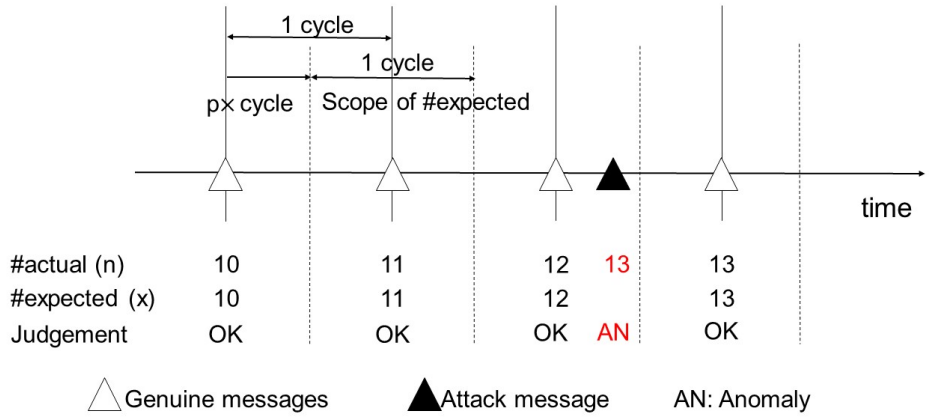


Figure 7.3: Concept of anomaly detection by cumulative sum

received messages are respectively 11 and 12. The timing of the intervals recovers when the expected number of received messages is 13. In this example, the attack is injected immediately before the recovery from the delayed message. The number of actually received messages $n$ exceeds $x$ when $x$ is 13, and the situation is detected as an attack. On the other hand, when the attack is not injected in the same situation, there are no false positives because $n$ is smaller than $x$ until the intervals recover from the delay.

Reliable detection is possible using the proposed method even when the detected time of the attack is not exactly the same as the actual attack time. Many of the existing methods have the same characteristic. Therefore, we will allow this characteristic.

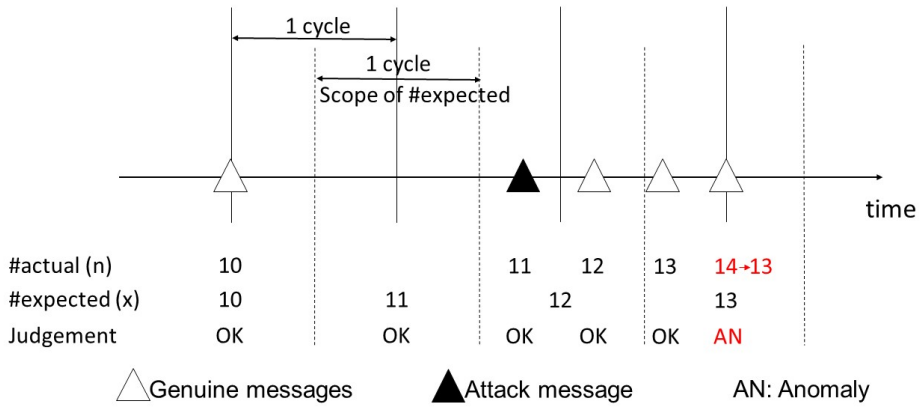


Figure 7.4: Behavior of anomaly detection by cumulative sum in a delay situation (an attack is injected)

### 7.3.4 Early arrival of messages

The basic idea shown in Section 7.3.2 cannot be used when genuine messages arrive earlier than the scheduled cycles. In the proposed method, the judgement is postponed by 1 reception when $n = x+1$ by storing the information of $n = x+1$. In the next reception, by using $n$, $x$, and the stored information, the situation is judged as normal or abnormal. We use an early arrival flag as the stored information to detect an attack. When $n = x + 1$ due to the early arrival of a message, the flag becomes ON, and the situation is judged to be normal temporarily. After that, when $n = x + 1$ and the flag is ON when the messages are received, the situation is judged to be an attack. When the attack is not injected during an early arrival, $n \leq x$ is satisfied and the early arrival flag is set to OFF. When $n > x+1$ is satisfied, the situation is judged as abnormal immediately. The algorithm is described in

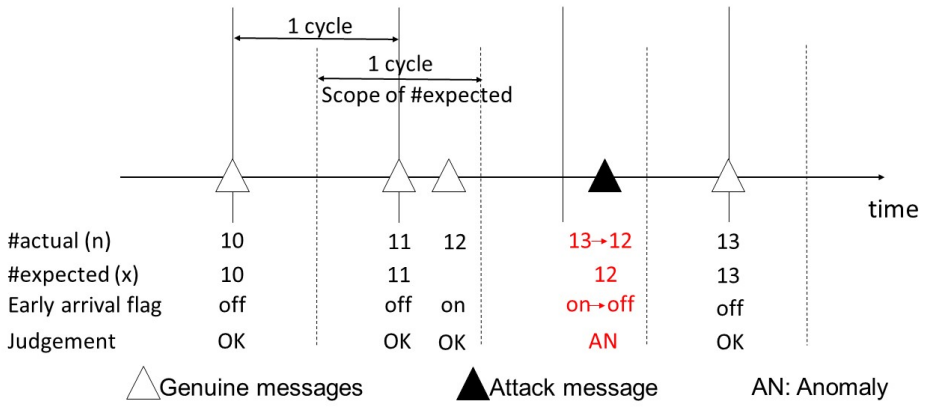


Figure 7.5: Behavior of anomaly detection by cumulative sum in an early arrival situation (an attack is injected)

detail in Section 7.3.6.

Figure 7.5 shows an example of ADCS for an early arrival. In this example, an early arrival occurs when the expected number of received messages is 11, $n = x + 1$ is satisfied, and the early arrival flag is set to ON. Thereafter, $n = x + 1$ is satisfied when the expected number of received messages is 12, the early arrival flag is ON, and the situation is detected as an anomaly. On the other hand, in the same situation without an attack, although the early arrival flag is set to ON when $n = 12$, the flag is set to OFF when $n = 13$. The detector judges the message as genuine. This judgment causes no false positives.

To allow two or more early arrivals, the start timing of the early arrival must be stored with an early arrival flag. If a number of early arrivals $e$ is allowed, the situation recovers to normal $e$ cycles after the start timing. If the following three

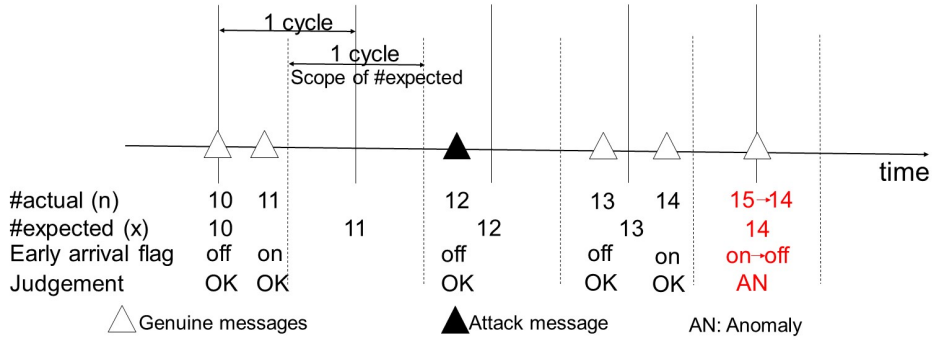


Figure 7.6: Behavior of anomaly detection by cumulative sum in delayed and early arrival situations (an attack is injected)

conditions are all met when a message is received, the situation is detected as an anomaly.

1. $x < n \leq x + e$ is satisfied.

2. The early arrival flag is 'ON'.

3. $n >$ 'the early arrival start-timing' $+e$ is satisfied.

### 7.3.5 Combinations of delayed and early arrival messages

Even when delays and early arrivals both occur, there are almost no false positives or negatives when the idea explained in Section 7.3.4 is used. Figure 7.6 shows an example of ADCS when an attack is injected with both delayed and early arriving messages. In this example, the attack is injected when the expected number of received messages is 12. The attack is detected when the expected number of

received messages is 14.

### 7.3.6 Proposed method

The proposed method can only be used for messages whose CAN-IDs are the same. Therefore, some CAN-IDs of the monitoring target are determined before the proposed method is used. The method is applied independently to targets whose CAN-IDs are different. In ADCS, the following expression is evaluated at the reception timing of the $i$-th message $t_i$ ($i = 0, 1, 2, ...$):

$$N_i - X_i > e \tag{7.1}$$

$$N_i - X_i > 0, N_i - X_i \leq e \tag{7.2}$$

where $N_0 = 0$, $X_i = \lfloor \frac{t_i - t_0 + p \times T}{T} \rfloor$.

$X_i$ expected number of received messages

$N_i$ number of actual messages

$e$ allowable early arrival

When a message is received, $N_i = N_{i-1} + 1$. After that, the situation is judged as 'Normal' or 'Abnormal' (Anomaly) depending which expression is satisfied.

1. If expression (7.1) is satisfied, the situation is judged as 'Abnormal'.

2. If expression (7.2) is satisfied, the situation is judged as 'Normal' temporarily. After that, if it continues for $e$ cycles after expression (7.2) is satisfied, the judgement changes to 'Abnormal'.

3. Otherwise, the situation is judged as 'Normal'.

If the $i$-th message is judged as 'Abnormal', $N_i = N_i - 1$.

### 7.3.7 Detection capabilities of anomaly detection by cumulative sum and the existing methods

In the existing methods and proposed method, the timings for anomaly detection are not exactly the same as the reception timings of the attack messages. Therefore, the number of false positives and negatives derived from timing inconsistencies between them is not a useful measure for a comparison of accuracies. Instead, we simply define a false positive as the exceeding number of detections against attack messages.

#### Cycle detection

In this method, genuine messages are incorrectly detected as attack messages when long delays and early arrivals significantly deviate the reception cycle of genuine messages; namely, when the deviation exceeds the permissible boundary. Moreover, this method may overlook an attack when the permissible boundary is

too wide. Therefore, the parameter for adjusting the permissible boundary is very important. This method cannot detect attacks correctly in a reception deviation situation.

**Delayed-decision cycle detection**

In this method, a false negative happens when an attack message is received just after a genuine message $M_{g1}$ and reception of the next genuine message $M_{g2}$ is delayed. More specifically, the delay is that interval between $M_{g1}$ and $M_{g2}$ exceeds $T + \beta$. Moreover, the method of determining $\alpha$ and $\beta$ is not described. These parameters should be tuned by hand for each CAN-ID, which is time consuming. This method can detect attacks correctly in the case of a small and medium reception deviation. The detection timing may be delayed compared with actual timing in some situations.

**Waszecki's method**

In this method, the worst-case value of jitter is very important for accurate detection. If the worst-case jitter $j$ is not set correctly, false positives or negatives may occur. Moreover, when the worst-case jitter is more than 1 cycle, we cannot derive correct values for the other parameters, especially $\delta$, because $\delta$ is derived from the greatest common divisor (GCD) of the cycle $T$ and $T - j$. When $T < j$, the value must be derived from the GCD of $T$ and $T - j$ whose sign is negative.

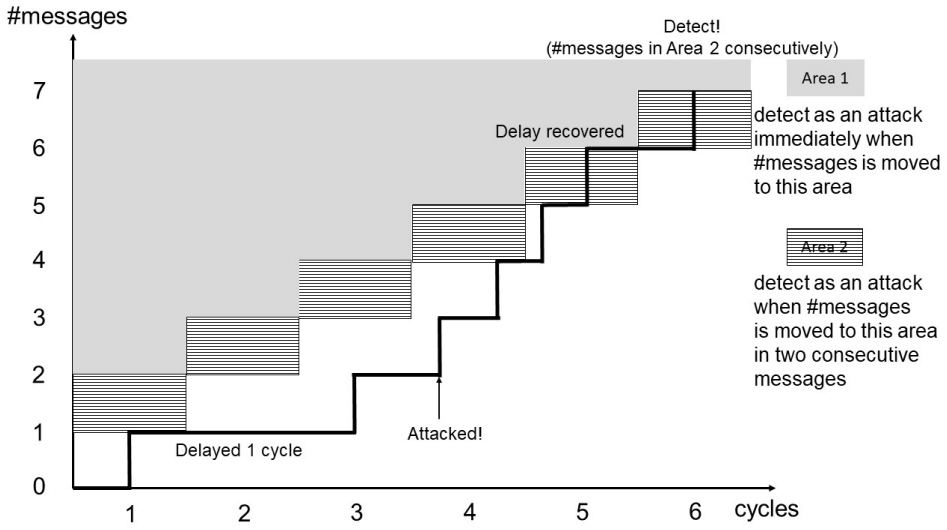


Figure 7.7: Example of anomaly detection by cumulative sum in a delayed situation

Thus, we think that their method cannot detect attacks accurately when the jitter is large. When the parameters are adjusted correctly, this method can detect attacks correctly in the case of a small, medium, and large (< 1 cycle) reception deviations. The parameter adjustment is not easy in some cases. The detection timing may be delayed compared with actual timing in some situations.

**Anomaly detection by cumulative sum**

An example of ADCS is shown in Figure 7.7 and Figure 7.8. In these figures, if the number of cumulative sums of actually received messages (NCA) falls into Area 1, ADCS judges the situation to be an attack immediately. Moreover, if the NCA falls into Area 2 for two consecutive messages, ADCS judges the sit-

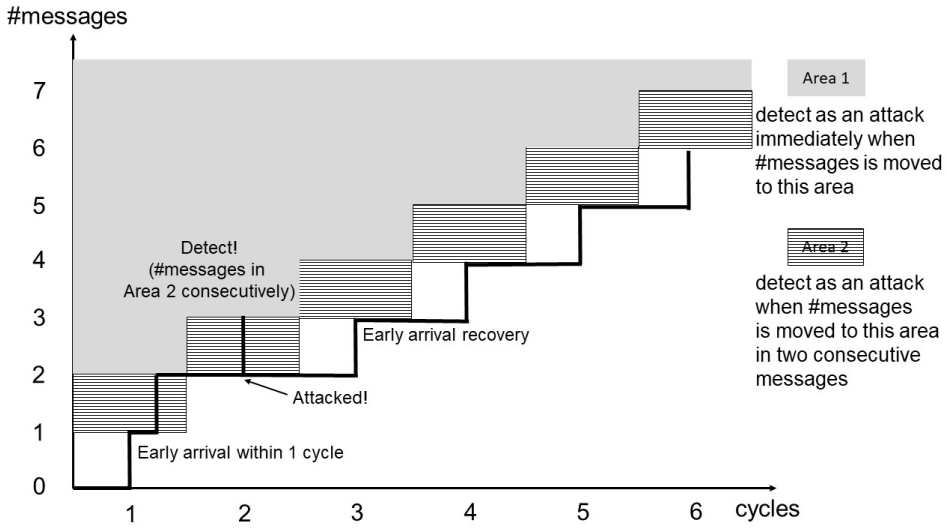


Figure 7.8: Example of anomaly detection by cumulative sum in early arrival situation

uation as an attack. After that, NCA is decremented by 1. On the other hand, in Waszecki's method, if NCA falls into Area 1 or Area 2 even just once, the situation is immediately detected as an attack. Note that they didn't show the method of decrementing NCA after the detection. We will assume that all genuine messages are received. Therefore, after recovery from a large delay and/or early arrival, the NCA returns to the expected value. When an attack is injected in a delayed and early arrival situation, the attack will be detected immediately after the delayed and early arrival situation recovers to normal. In our method, the maximum allowable number of early arrivals $e$ is important. This parameter signifies the maximum number of early arriving messages within $e$ cycles. When the

number of early arrivals or the number of recovery cycles exceeds $e$, a false positive occurs. However, we can empirically set $e$ to be large enough because there is very little possibility of a great increase in early arrivals. Actually, we found that setting $e = 1$ caused no false positives in many time evaluations. $p$ means the boundary of the counting parameter. In our method, the expected number of received messages is increased by 1 every cycle without depending on $p$. Therefore, we can choose $p$ empirically. When $e$ is set correctly, this method can detect attacks correctly regardless of whether the reception deviation is small, medium, or large. We think that the detection accuracy is highest for periodic messages when the boundary of the count is set to half of the reception cycle. Moreover, it is easy to adjust the parameters $e$ and $p$ in all cases. The detection timing may be delayed compared with the actual timing in some situations.

As summarized in Table 7.1, our discussion shows that the ADCS is the best among the four detection methods.

Table 7.1: Comparison of existing methods and proposed method

| | Cycle[10] | Delayed-decision Cycle[11] | Waszecki [48] | Anomaly Detection by Cumulative Sum (proposal) |
|---|---|---|---|---|
| Small deviation on cycle[1] ($\leq$ 16.6%) | Good | Good | Good | Good |
| Medium deviation on cycle[1] ($\leq$ 50%) | Bad | Good | Good | Good |
| Large deviation on cycle[1] ($>$ 50%) | Bad | Not Good[3] | Fair[4] | Good |
| Parameter adjustment | Difficult | Difficult | Difficult | Easy |
| Real time characteristic[2] | Good | Fair[5] | Fair[5] | Fair[5] |

[1]This deviation is independent from the message priority (CAN-ID).

[2]This is the delay of detection from the actual received message.

[3]This means the DDC method seems not to be good in the case of a large deviation, but its detection ability seems to be better than that of the cycle method.

[4]This means the detection may cause a false negative when the worst-case jitter is over 1 cycle.

[5]This means the detection delay is a few cycles. However, we think this delay is not a major problem.

### 7.3.8 Implementation

#### Reinitialization of the counters and the first timestamp

In order to use ADCS effectively, it is preferable to reinitialize the counters $x$ and $n$. Because there may be a small gap between the installed cycle information for the detection algorithm and the actual cycle of the vehicle, false positives or negatives may occur for a long span of time when using ADCS. To solve this problem, reinitialization processing is needed. The processing involves setting $x = x - n$ and $n = 0$ and adjusting the counting boundary. In addition, this processing has variable execution conditions. Some of them are as follows.

1. Exceeding the predetermined number of receptions

   The reinitialization processing is executed when the number of received messages exceeds a predetermined threshold $r$. This condition is too simple because the attacker may learn the threshold value when the attacker send the attack messages at the reinitialization timings, false positives or negatives may occur.

2. Exceeding the randomly determined number of receptions

   The reinitialization processing is executed when the number of received messages exceeds a randomly determined threshold $r_j$. This method decreases the success probability of the attack described above.

3. Limited reception interval

   Here, the number of received messages exceeds a predetermined or randomly determined threshold. In addition, the reinitialization processing is executed when the interval between the last message and the one before the last is within another threshold $q$. This method dramatically decreases the success probability of the attack described above.

**Algorithms with Reinitialization**

Algorithm 1, 2, and 3 are implementations of the above methods. The parameters are shown in Table 7.2. $e$ is assumed to be 1, and the allowable recovery cycle for early arrivals $w$ is also assumed to be 1. It is easy to enhance the algorithm to $w(\geq e)$ at $e \geq 2$. The user of this algorithm should pre-determine the reinitialization parameter $r$, reinitialization condition $q$, and offset ratio of increment timing $p$. For example, we will suppose that these values are roughly $r = 10$, $q = 0.1$, and $p = 0.5$. The algorithm consists of a pre-computation, main function, and reinitialization processing. The pre-computation is executed one time upon first reception of a message for each CAN-ID. The main algorithm is executed every time a message is received with $i \geq 1$ for each message with the same CAN-ID.

Table 7.2: Parameters

| | |
|---|---|
| $i$ | order of message reception($i = 0, 1, 2, ...$) |
| $r_j$ | $j$-th number of received messages before reinitialization. (e.g. $r_j$=10) (pre-determined) |
| $p$ | parameter of the offset ratio of increment timing whereby $0 < p < 1$ (generally, $p = 0.5$ is preferable). (pre-determined) |
| $T$ | reception cycle of message (in general, this is indicated by the average cycle) (pre-determined) |
| $t_i$ | reception timestamp of the $i$-th message whose CAN-ID is $ID$ |
| $d$ | next expected reception time |
| $n$ | actual number of received messages |
| $x$ | expected number of received messages |
| $f$ | early arrival flag |
| $ID$ | CAN-ID |
| $J$ | Judgement result for each reception |

**Algorithm 1** Pre-computation of proposed method

**Input:** $p, T, t_0$

**Output:** $d, n, x, f$

$d = t_0 + (1 + p) \times T$

$n = 0, x = 1, f = off$

Output$(d, n, x, f)$

**Algorithm 2** Main function of proposed method

**Input:** $r_j, p, T, t_i (i \geq 1), d, n, x, f$

**Output:** $J, d, n, x, f$

$n = n + 1$

**while** $t_i > d$ **do**

    $x = x + 1, d = d + T$

**end while**

**if** $n \leq x$ **then**

    $J \Leftarrow$ 'Normal'

    $f = off$

**else**

    **if** $n = x + 1$ **then**

        **if** $f = off$ **then**

            $J \Leftarrow$ 'Normal'

            $f = on$

13: **else**

14: $J \Leftarrow$ 'Abnormal'

15: $n = n - 1$

16: $f = off$

17: **end if**

18: **else**

19: $J \Leftarrow$ 'Abnormal'

20: $n = n - 1$

21: $f = on$

22: **end if**

23: **end if**

24: Call the following "Reinitialization processing"

25: Output($J, d, n, x, f$)

**Algorithm 3** Reinitialization processing of proposed method

**Input:** $d, n, x, p, T, r_j, t_i, t_{i-1}, q$

**Output:** $d, n, x$

1: **if** $n \geq r_j$ **then**

2: **if then**$t_i - t_{i-1} < q$

3: $x = x - n, n = 0$

4: $d = t_i + p \times T$

5: **end if**

6: **end if**

# Chapter 8

# Proposal 2: Format Estimation

In this chapter, we describe a new method for estimating the format of the CAN message payload. This method was first proposed in [65], [66]. We call this method "Single Message Format Estimation (SMFE)".

## 8.1 Estimating the format of the field data in the CAN payload

### 8.1.1 Existing format estimation methods

#### Markovitz's method

In 2015, Markovitz et al. proposed a format estimation algorithm [12] that can identify three field-data types, namely, constant value, counter/sensor value, and

multi value. In order to detect anomalous messages with high accuracy, a detector must estimate the format of CAN payloads with the same CAN-ID. To estimate it, the estimator considers all combinations of the length and starting bit of the field data. These parameters correspond to the field boundary. Our boundary is defined as a boundary of consecutive field data. One example of boundaries are '-' of '(1-bit field-data)-(2-bit field-data)-(1-bit field-data)-(4-bit field-data)-(16-bit field-data)-(32-bit field-data)-(8-bit field-data)' for a 64-bit payload. This example is shown in Figure 8.1. Any combination can be represented by pairs of the starting bit $s$ and length $l$ of the field data. For this example, the equivalent representation is '(0,1)-(1,2)-(3,1)-(4,4)-(8,16)-(24,32)-(56,8)'. For the 64-bit payload, field data whose starting bit is 0 has a length that is one of $1, 2, \ldots, 64$, namely, possible patterns of field data length at position 0 is 64. For a field data whose starting bit is 1, field data whose length is 64-bit cannot be stored because remaining length of the payload is 63. So, possible patterns of field data length at position 1 is 63. Similarly, patterns of field data length for each position can be calculated. Totally, there are $64 + 63 + 62 + \ldots + 1 = 2080$ patterns for a 64-bit payload.

Their method picks position and length pairs from the above 2080 patterns and calculates the field-data type and score by using the number of unique values in the CAN log with the same CAN-ID for the extracted pair. After that, an optimal combination of pairs that may have duplicated positions and lengths are selected

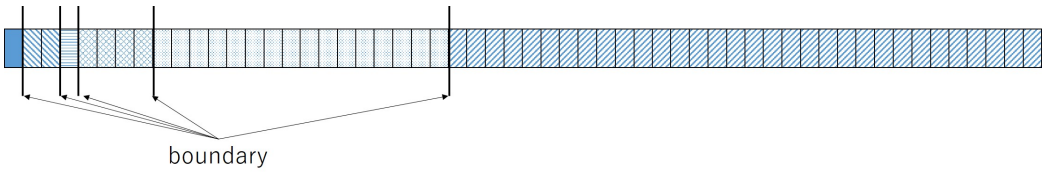


Figure 8.1: Examples of boundaries

depending on the type and score.

**Kishikawa's method**

In 2017, Kishikawa et al. proposed an extension of Markovitz's format estimation [13]. Kishikawa's method can identify five field-data types, i.e., constant value, counter value, sensor value, check-sum, and multi value. In their method, the variance of the field data changes is used in addition to the number of unique values. When the change of the field data in the received CAN payloads is small, the variance is small as well. Their method classifies the sensor value and check-sum value by using the variance. It selects pairs from duplicate positions and lengths by using priority. The priority order is constant value followed by counter value, sensor value, check-sum value, and multi value. Compared with Markovitz's method, Kishikawa's method can identify more types of field data. Therefore, its detection accuracy is expected to be higher than that of Markovitz's method.

#### Stone's method

In 2019, Stone proposed a field-data boundary estimation method [14]. In this method, the number of bit-flips is counted at each bit. After that, the field-data boundaries are derived by using a hill-climb algorithm. This method is rather different from the other methods. It estimates boundaries correctly when the format of the CAN payload is simple. We think that it cannot estimate the boundaries correctly when a complex payload is given.

#### Other format estimation methods

Recently, a number of methods have been developed for estimating the CAN format with high accuracy [67], [68], [69].

## 8.2 Proposed method (single message format estimation)

The methods described in Section 8.1.1 cannot identify event-based messages. In addition, there are more field-data types in actual CAN logs that cannot be identified with these methods. We improved Kishikawa's method in order to identify more field-data types. As described below, our method can identify five new field-data types.

### 8.2.1 Basic idea

Our method is based on Markovitz's method and Kishikawa's method. To achieve high estimation accuracy, the main differences between our method and these methods are as follows.

- Our method has five new field-data types and a corresponding priority order.
- The reception time interval (for event based messages), a power of 10 (for A-Value), the sign of the data, and differential of the data are used to estimate the new field-data types.

Our algorithm uses 2080 windows at maximum (depending on the message length which is determined by the CAN-ID) for all messages which have the same CAN-ID. "Position" means the start bit of one window in the CAN message, and "Length" means the length of the window. The representations of the position and length are illustrated in Figure 8.2. First, we check for the possibility of each type defined in Section 8.2.2 in each window of the CAN payload. In particular, all pairs of position and length are checked. Then, the unlikely pair types are excluded from the candidates. After that, we determine the type according to the following priority: A-Value, Constant, S-Event, C-Event, Counter, Increment, S-Sensor, U-Sensor, Checksum, and Others. If two or more pairs have the same priority type, the longer one is chosen. A small example of format estimation is illustrated in Figure 8.3. In this example, 3-bit CAN data with the same CAN-ID is received.

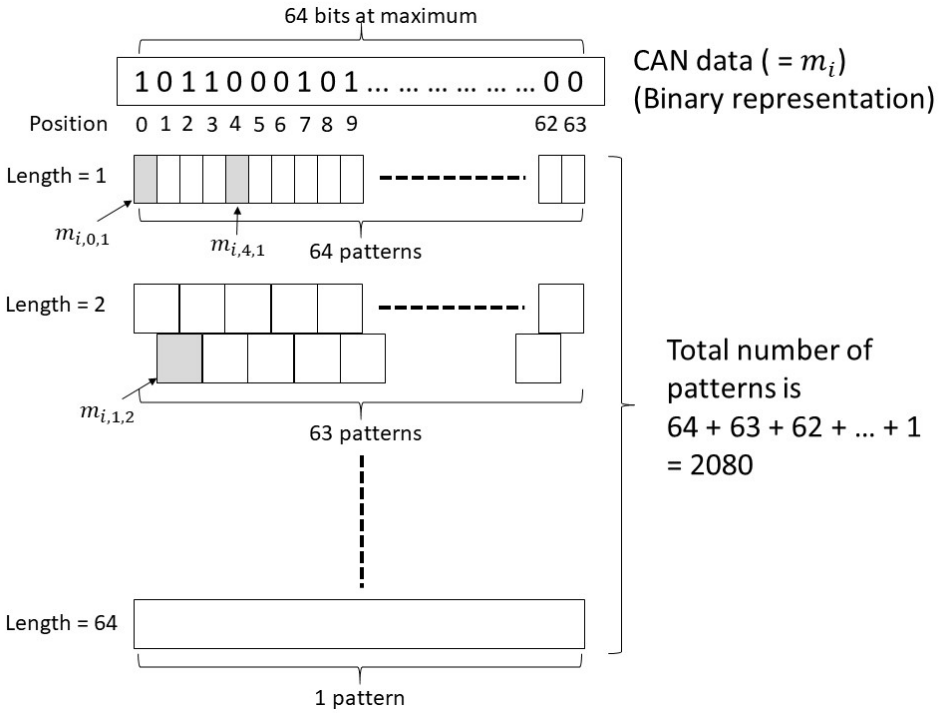


Figure 8.2: Representation of position and length

First, windows are prepared for all positions and lengths. After that, the possibility of each field type is evaluated in each window when a new message is received. The result of the evaluation is the estimated format of the received CAN data.

### 8.2.2 Field-data types

Our method can estimate ten field-data types and boundaries. Although the actual number of field-data types has not been published, we think that by checking for five more types than the previous methods check for, our method achieves higher estimation accuracy.

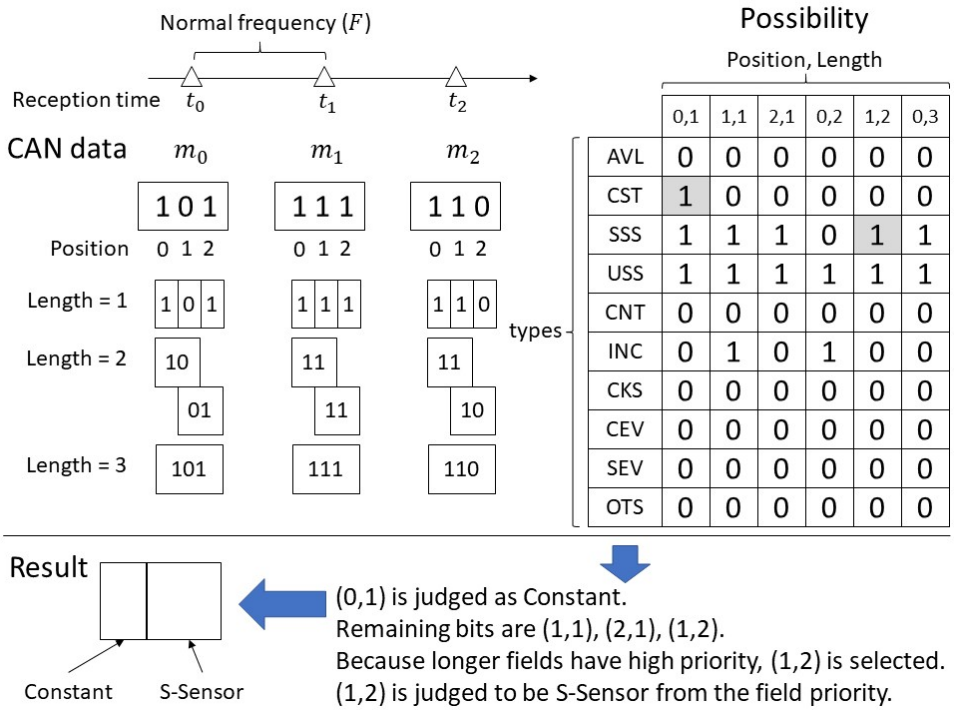


Figure 8.3: Small example of format estimation

1. Constant

   Defined as field data that has a fixed value, this type is estimated by the variation in each data.

2. S-Sensor (Signed Sensor Value)

   Defined as signed field data that may change slightly within a period, this type is estimated by the variance of each data differential.

3. U-Sensor (Unsigned Sensor Value)

   Defined as unsigned field data that may change slightly within a period, this type is estimated by the variance of each data differential.

4. Counter

   Defined as changing field data whose differential between two consecutive data is constant, this type is estimated by each data differential.

5. Increment

   Defined as changing field data whose differential between two consecutive data is arbitrarily small, this type is estimated by each data differential.

6. Checksum

   Defined as checksum field data whose value is calculated in the specified manner or is distributed over all possible values over a long time, this type

is estimated by the manner of the derivation of the checksum determined by vehicle type.

7. C-Event (value changes when an event occurs)

   Defined as field data whose value always changes when an event occurs and does not change when events do not occur, this type is estimated by the message reception interval and value of field data.

8. S-Event (value takes on a specific value when an event occurs)

   Defined as field data that takes on a specific value when an event occurs and take another value when events do not occur, this type is estimated by the message reception interval and value of field data.

9. A-Value (near the appropriate values)

   This type is estimated by the value of the field data. An appropriate value is one that it is likely to be meaningful in some pre-determined sense (it is determined in a heuristic manner). We set these values as powers of 10, integer multiples of a power of 10, and their combinations.

10. Others

    This type is for field data that are not of any of the above nine types.

### 8.2.3 Algorithm

Our algorithm is composed of three sub-algorithms: Algorithm 4, 5, and 6. Algorithm 4 is the parent algorithm of Algorithm 5 and 6. Algorithm 5 extracts field-type candidates at each position and length. First, all types are marked as candidates. This algorithm eliminates all types that have no possibility of being a candidate. Algorithm 6 determines the type from the candidates. In this determination, all types that have duplicate positions in the payload are evaluated in terms of the length and priority order. As mentioned above, the appropriate value is defined as a power of 10. All pairs of position and length are considered. The possibility of each field-data type is represented by possibility flags (0: no possibility, 1: possibility remains); for example, the possibility flag of a constant field is denoted as "$CST$". The possibility flags of the other type are denoted in the same manner, i.e., $AVL$ (for A-Value), $SSS$ (for S-Sensor), $USS$ (for U-Sensor), $CNT$ (for Counter), $INC$ (for Increment), $CKS$ (for Checksum), $CEV$ (for C-Event), $SEV$ (for S-Event), and $OTS$ (for Others).

**Algorithm 4** Format Estimation Algorithm (Proposed)

**Input:** CAN log including at least one target message $M$, time-stamp $T$ of reception of $M$, message length *mlen*, frequency $F$ (for periodic messages).

**Output:** Format is a group of $field$s: Each $field$ includes information of the form (*type*, *position*, *length*).

1: Set possibility flags $AVL_{pos,len,k} = 1$, $CST_{pos,len} = 1$, $SSS_{pos,len} = 1$, $USS_{pos,len} = 1$, $CNT_{pos,len} = 1$, $INC_{pos,len} = 1$, $CKS_{pos,len} = 1$, $CEV_{pos,len} = 1$, $SEV_{pos,len} = 1$, $OTS_{pos,len} = 1$, and $i = 1$

2: $m_0 \leftarrow$ top of $M$, $t_0 \leftarrow$ time-stamp of reception $m_0$, $m_0$ is excluded from $M$

3: Call *CandidateExtraction*(Algorithm5)

4: Call *CandidateSelection*(Algorithm6)

**Algorithm 5** Candidate Extraction (Proposed)

**while** $M \neq \phi$ **do**
  $m_i \leftarrow$ top of $M$, $t_i \leftarrow$ time-stamp of reception $m_i$, remove $m_i$ from $M$
  **for** $len = 1$ to $mlen$ **do**
    **for** $pos = 0$ to $mlen - len$ **do**
      **for** maximum $k$ such that $10^k \leq 2^{mlen}$ to 1 **do**
        **if** NOT $m_{i,pos,len} \approx 10^k$ **then**
          $AVL_{pos,len,k} = 0$
        **end if**
      **end for**
      **if** $m_{i,pos,len} = m_{0,pos,len}$ AND $t_i - t_{i-1} \neq F$ **then**
        $SEV_{pos,len} = 0$, $CEV_{pos,len} = 0$
      **else**
        **if** $m_{i,pos,len} \neq$ Specific value **then**
          $SEV_{pos,len} = 0$
        **end if**
      **end if**
      **if** $m_{i,pos,len} \neq m_{0,pos,len}$ **then**
        $CST_{pos,len} = 0$
      **else**
        **if** $m_{i,pos,len} - m_{i-1,pos,len} \neq m_{i-1,pos,len} - m_{i-2,pos,len}$ **then**

21: $CNT_{pos,len} = 0$

22: **if** $(m_{i,pos,len} - m_{i-1,pos,len}) \times (m_{i-1,pos,len} - m_{i-2,pos,len}) < 0$ **then**

23: $INC_{pos,len} = 0$

24: **end if**

25: **end if**

26: **end if**

27: **if** $m_{i,pos,len}$ is not valid for checksum value **then**

28: $CKS_{pos,len} = 0$

29: **end if**

30: **end for**

31: **end for**

32: **end while**

33: **if** variance of $\Delta m_{eachposition,pos,len} = 0$ **then**

34: $AVL_{pos,len,allk} = 0$

35: **else**

36: **if** variance of $\Delta m_{eachposition,pos,len} > thresholdvalue(big)$ **then**

37: $AVL_{pos,len,allk} = 0$, $SSS_{pos,len} = 0$(considered by signed value), $USS_{pos,len} = 0$(considered by unsigned value)

38: **end if**

39: **end if**

**Algorithm 6** Candidate Selection (Proposed)

1: **for** $len = mlen$ to 1 **do**

2: **for** $pos = 0$ to $mlen - len$ **do**

3: **if** $AVL_{pos,len,k} = 1$ for bigger $k$ **then**

4: $field_{pos,len} = A - Value_k, pos = pos + len$

5: **if** $CST_{pos,len} = 1$ **then**

6: $field_{pos,len} = Constant, pos = pos + len$

7: **end if**

8: **end if**

9: **end for**

10: **end for**

11: In the same manner, $field_{pos,len}$ is set in the following pre-determined order of priority (SEV, CEV, CNT, INC, SSS, USS, CKS, and OTS)

# Chapter 9

# Proposal 3: Data Relation Analysis

In this chapter, we introduce our relational analysis for multiple CAN messages. This method was proposed in the domestic conference [70] and the 22nd International Conference on Network-Based Information Systems (NBiS) [71]. We call this method "Multiple Messages Relation Analysis (MMRA)". We are applying for a patent of this method [72].

## 9.1 Data relation analysis

This method finds static relations between the field data in data fields by observing their dynamic behavior. We focus on messages whose CAN-IDs are different from each other. We evaluate values of field data in the CAN data. In the extraction phase described in Section 6.1.1, the number of times to match a specific value is

derived for each field data. The number of changes of each value is also derived for each field data. Additionally, the timings when the value of field data matches the specific value or the timings when the value changes are evaluated. The data relation analysis uses the following procedure. Note that the field data is assumed to be at a possible position and be of a possible length in a message.

**General algorithm**

1. For each data, evaluate the timing whose interval of consecutive reception is deviated from the normal transmission interval (or the transmission timing for non-periodic messages.)

2. Using the timings gathered in step 1, derive the number of times the value changes and the number of times the value matches a specific value.

3. Derive all data pairs in step 1 that have the same timings and the same number derived in step 2.

Step 1 of the above algorithm is difficult because the message reception interval is evaluated by using the permissible boundary. Specifically, when we evaluate sameness of two reception timings of different messages, permissible boundary for each timing of message reception is calculated, and sameness are judged by using the permissible boundaries. After that, if two consecutive timings are judged as the 'same', the interval of them is calculated, and the deviation of interval is

evaluated by using permissible boundary for intervals. Therefore, permissible boundaries are used at 2 phases. Using of permissible boundaries in 2 phases cause rough approximation of the actual deviation. So, the above algorithm has following problem.

**Problem of the general algorithm**

1. The evaluation of the deviation is inaccurate because it uses the permissible boundary in two phases.

In the next section, we propose an algorithm that does not have this problem.

## 9.2 Proposed method (multiple messages relation analysis)

### 9.2.1 Overview

To solve the problem of the general algorithm, we reduce the number of evaluations that use the permissible boundary. The essence of the problem is in the check of whether there are deviations from the normal interval. The following algorithm judges whether the reception interval between two messages is nearly equal to that of another pair of messages by using one permissible boundary. The timing deviation can be evaluated more accurately by reducing the number of evaluations

that use the permissible boundary.

**Algorithm 7** Proposed Algorithm

1: For each data, derive the number of times the value changes or the number of times the value matches a specific value.

2: For each data, derive the message reception interval.

3: Derive all field data pairs that have the same number in step 1 and the same intervals in step 2 (using the permissible boundary).

### 9.2.2 Each step of algorithm 7

#### Step 1

The length of the payload of each message can be derived from the CAN data. Generally, one payload contains multiple field data. However, we don't know how many field data are stored or how the payload is divided up. First, we will describe how to count the number of changes of field data because it is simpler than counting the number of times of matches a specific value.

We prepare counters for all positions which have the possibility of field data being stored there. For example, the maximum length of a CAN message is 64 bits. For the 64-bit payload, field data whose starting bit is 0 has a length that is one of $1, 2, \ldots, 64$, namely, possible patterns of field data length at position 0 is 64. For a field data whose starting bit is 1, field data whose length is 64-bit cannot

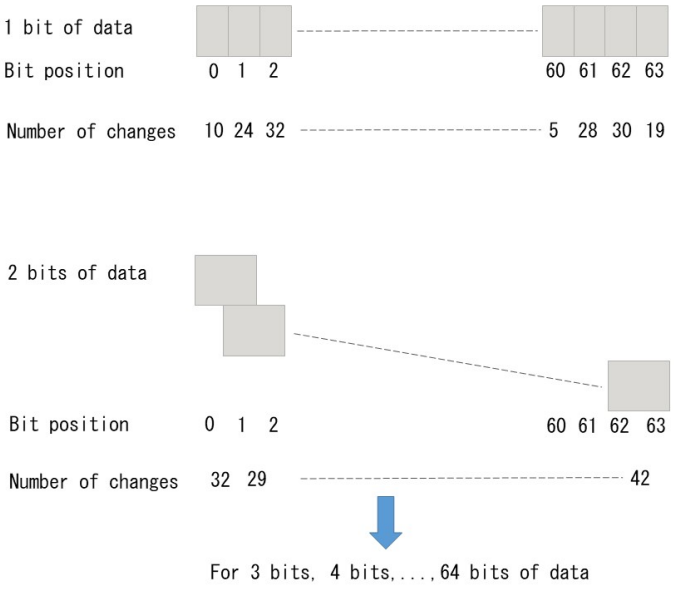


Figure 9.1: Overview of counting procedure

be stored because remaining length of the payload is 63. So, possible patterns of field data length at position 1 is 63. Similarly, patterns of field data length for each position can be calculated. In total, there are $64 + 63 + 62 + \ldots + 1 = 2080$ patterns, so we prepare 2080 counters for each. After counters have been prepared, we count the number of changes to another value for each field data length at each position. The number of changes is counted for each field data length at each position in a message which has the target CAN-ID, while the CAN log is read from the top of it. An overview of this counting procedure is shown in Fig. 9.1.

Next, we describe the method of counting the number of times that the value matches a specific value. Here, we prepare counters at each length, position, and value. The patterns are derived from the data length. For example, when data

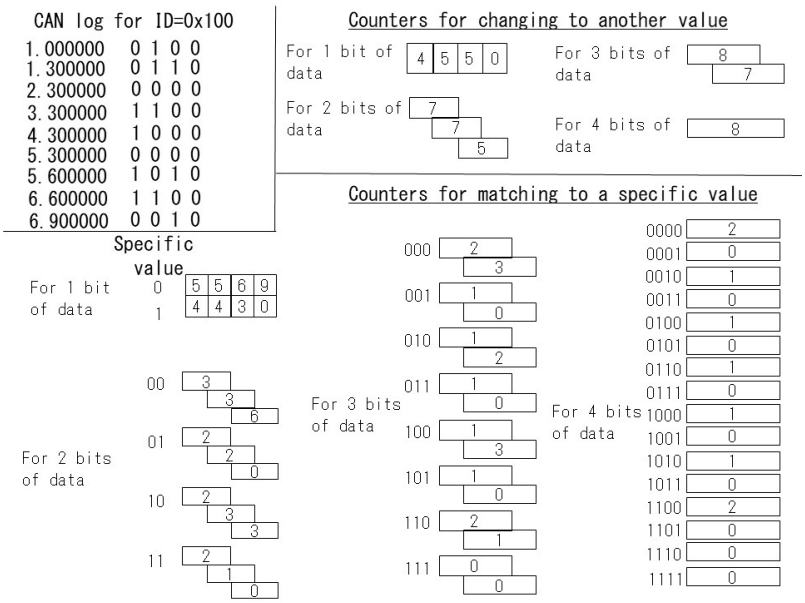


Figure 9.2: Example of counting

length is 2 bits, there are 4 patterns, namely 00, 01, 10, and 11. For 2-bit data, there are 63 patterns. Therefore, $4 \times 63 = 252$ counters are needed for 2-bit data. However, for example, there are $2^{64}$ patterns for 64-bit data, meaning that we cannot prepare all counters for long data because it would exceed the available memory of the PC. To solve this problem, we prepare counters corresponding to the values that appear in the CAN log. Figure 9.2 shows examples of counting the number of matches each prepared value. For ease of understanding, this example is only for 4-bit data, which is not allowed in the CAN specification. In Fig. 9.2, we don't prepare counters for the value is '0'.

**Step 2**

In this step, message intervals are derived. The intervals between timings of changes of the value of field data are derived. Similarly, the intervals between timings of matching of field value to a specific value and the next matching to the specific value are derived.

**Step 3**

This step derives the field data relations from the results of steps 1 and 2. First, field data on the number of changes to another value or matches a specific value are collected from the results of step 1. A set $S_i$ is composed from the counter whose number is $i$. Next, by using the results of step 2, field data whose timing of first change to another value or whose timing of first matches the specific value are the same are collected from each element of $S_i$. Field data whose intervals of changes to another value or intervals of matches the specific value are the same are selected from the collected data. The collected data are composed as $S_{i,j}, S_{i,j+1}, \ldots$ by number and timing. Finally, the related data $S_{i,j}, S_{i,j+1}, \ldots$ are output.

# Chapter 10

# Evaluations

## 10.1 Evaluation of method 1

In this section, we evaluated detection accuracy of the proposed method 1.

### 10.1.1 Overview

To confirm the properties shown in Table 7.1, we evaluated the detection accuracy of each method by simulating the sending and receiving of CAN communications on a PC. We obtained genuine data from an actual vehicle and injected many attack messages into them. We examined the accuracies of the attack detection methods.

Table 10.1: Types of genuine data

| Name | Length | Cycle | Features |
|---|---|---|---|
| Data 1 | 600 sec | 9999 microseconds | Few delayed arrivals and early arriving messages. |
| Data 2 | 600 sec | 9982 microseconds | Many delayed arrivals and early arriving messages. |

### 10.1.2 Data preparation

We drove a car and recorded CAN logs from the OBD-II port for over ten minutes. After that, we looked for perfectly periodic messages and quasi-periodic messages in the log. We found variations in the genuine data and decided to use two (Data 1 and Data 2 in Table 10.1): Data 1 is a perfectly periodic message and Data 2 is a quasi-periodic message. We think that both types of message can be evaluated (they are the detection targets of our method). The data had mutually different CAN-IDs. The data (no attack injected) was first evaluated to confirm whether they cause false positives. Next, we injected attack messages into Data 1 and Data 2 and checked whether false positives or negatives occur. Method 1 used only the message reception timing, not the field data. Therefore, we could use any value for content of the attack messages.

In the data preparation task, we made many attack-data based on the three variations shown in Table 10.2. The first type was a single attack in which an

attack message is injected at a random timing into one genuine data. We prepared 10000 data of this attack for each genuine data. In the second type of the attack, 100 attack messages were injected into one genuine data. Each attack message was injected at a random timing. We prepared 10000 data of this attack type for each genuine data. In the third type of the attack, 100 periodic attack messages were injected to one genuine data. The injecting timing of the first attack message was determined at random. We prepared 10000 data of this attack for each genuine data.

Table 10.2: Patterns of attack data

| Name of pattern | Number of dataset | Features of attack message(s) |
|---|---|---|
| Pattern 1 | 10000 | 1 message with random injection times in one dataset |
| Pattern 2 | 10000 | 100 messages with random injection times in one dataset |
| Pattern 3 | 10000 | 100 messages at the appropriate cycle in one dataset. The 1st message is injected at a randomly chosen time. |

### 10.1.3 Parameter adjustment

We adjusted the parameters of the three existing methods and our method 1.

#### Cycle detection

There are a number of cycle detection methods. We chose the method [10] described in Section 7.2.1. It gives $\pm 1ms$ as the permissible boundary of cycle detection. Using this parameter, the evaluation of Data 1 (no attack is injected) detected a large number of false positives (more than 60,000 messages in 600 seconds). Thus, we modified the specifications of the cycle detection so that the permissible boundary would always be updated when a message is received, even if the reception time is not within the permissible boundary. As a result, the number of false positives decreased considerably. We initialized the parameter of the permissible boundary with a very low value and repeated experiments using consecutively slightly higher values until the false positives disappeared. After conducting many experiments using this method, we decided to set the parameter of the permissible boundary to $\pm 8\%$ for Data 1 and $\pm 42\%$ for Data 2.

#### Delayed-decision cycle detection

Recommended parameters for the delayed-decision cycle detection are not provided in [11]. Therefore, deciding the parameter value is very difficult. This detection method consists of two phases, namely the selecting phase for selecting

input messages to the anomaly checking phase using $\alpha$ and the anomaly checking phase using $\beta$. At first, we thought that the detection capability seemed to be best when $\alpha$ was 0, because it means that the selecting phase using $\alpha$ can be passed. However, when we set these parameters, many false positives occurred in the experiments. The results are listed as "Delayed-decision cycle detection 1" in Table 10.3. Instead, we adjusted the parameters through trial and error experiments, finally setting $\alpha$ to 8% in the reception cycle for Data 1 and 42% in the reception cycle for Data 2. After that, we set $\beta$ to 92% for Data 1 and 61% for Data 2. The results are listed as "Delayed-decision cycle detection 2" in Table 10.3.

**Waszecki's method**

The method of deriving the worst-case jitter is not described in detail in [48]. We think that this value can be derived by using the following formula (1) ($T$ is the reception cycle.) In this case, the worst-case jitter is the maximum difference between the message interval and message reception cycle.

1. $j = \max_i(|T - (t_i - t_{i-1})|)$

By using the above formula, we derived $j = 72(usec)$ for Data 1 and $j = 4144(usec)$ for Data 2. However, many false positives occurred in the evaluation for Data 1. The result is listed as "Waszecki's method 1" in Table 10.3. Instead, we used another formula for $j$:

2. $j = \max_i(|t_i - (t_0 + i \times T)|)$

The above formula (2) means that the worst case jitter is the maximum difference between the reception time and the expected normal reception time derived from the first reception time. Formula (2) gave $j = 5062(usec)$ for Data 1 and $j = 5154(usec)$ for Data 2. The method worked well with these values. The results are listed as "Waszecki's method 2" in Table 10.3.

**Anomaly detection by cumulative sum**

Running ADCS on both Data 1 and Data 2 yielded no false positives when the allowable early arrival $e$ was set to 1. $p$ was set to 0.5 empirically. We chose (1) in Section 7.3.8 as the reinitialization method and used 20 for the threshold $r_j$ for all $j$.

### 10.1.4 Evaluation results

The results of the evaluation are shown in Table 10.3. In the cycle detection, up to two detections occur for one attack injection. Therefore, we cannot judge whether the cycle detection is correct because we cannot know how many detections occur for the attacks. In the delayed-decision cycle detection, the difficult parameter adjustment gives better results than those using the simple parameter decision, although the results are still not optimal. On the other hand, the number

of detections equals the number of attack injections for all datasets in the case of Waszecki's method with a well-optimized parameter and in ADCS. Therefore, these methods can detect attacks with high accuracy.

Table 10.3: Results of evaluation

| Data | Attack pattern in Table 10.2 | Number of total genuine messages (NA) | Number of total attack messages | Number of messages detected as anomalous (ND) | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Cycle | Delayed-decision cycle 1 | Delayed-decision cycle 2 (optimized) | Waszecki's method 1 | Waszecki's method 2 (optimized) | Cumulative Sum (proposal) |
| Data 1 | Pat. 1 | 676000000 | 10000 | 18406 | 13612 | 10000 | 82394523 | 10000 | 10000 |
| | Pat.2 | 676000000 | 1000000 | 1839102 | 1360543 | 999972 | 83390855 | 1000000 | 1000000 |
| | Pat. 3 | 676000000 | 1000000 | 1842671 | 1003663 | 1000000 | 83267634 | 1000000 | 1000000 |
| Data 2 | Pat. 1 | 706700000 | 10000 | 11641 | 10901 | 9988 | 10000 | 10000 | 10000 |
| | Pat. 2 | 706700000 | 1000000 | 1160431 | 1085404 | 997810 | 1000000 | 1000000 | 1000000 |
| | Pat. 3 | 706700000 | 1000000 | 1161273 | 1000802 | 998348 | 1000000 | 1000000 | 1000000 |

From the definition given in Section 7.3.7 gives the number of false positives when $N_{DA} = ND - NA > 0$ and the number of false negatives when $N_{DA} < 0$, $-N_{DA}$.

### 10.1.5 Discussion

**Comparison between anomaly detection by cumulative sum and Waszecki's method**

The detection accuracies of ADCS and the well-optimized Waszecki's method were the same in the evaluation. The discussion in Section 7.3.7 indicates that these methods may cause false positives or negatives in specific situations. To confirm this, we performed two small additional evaluations by creating two data whose reception cycle was 1 second. The data for these additional evaluations are shown in Figure10.1. Data 3 of the first additional evaluation included a large early arriving message whose jitter exceeded one cycle in Waszecki's method. The results indicated that ADCS caused no false positives or negatives, whereas Waszecki's method caused 5 false negatives, but no false positives. Data 4 of the second additional evaluation included two early arriving messages that exceeded the allowable early arrival threshold $e$ in ADCS. In this case, ADCS caused 1 false positive and no false negatives, while Waszecki's method caused no false positives and 1 false negative. This situation also caused a large jitter for Waszecki's method. Therefore, we think that ADCS was a little better in its detection accuracy compared with Waszecki's method.

| Reception time of Data 3 | Reception time of Data 4 |
|---|---|
| 11.000000 | 11.000000 |
| 12.000000 | 12.000000 |
| 13.000000 | 13.000000 |
| 14.000000 | 14.000000 |
| 15.000000 | 15.000000 |
| 15.800000 (early arrival, relative to 16.00) | 15.800000 (small deviation) |
| 15.900000 (very early arrival, relative to 17.00) | 15.900000 (1 early arrived) |
| 17.000000 (attack) | 16.100000 (2 early arrived) |
| 17.100000 (attack) | 16.200000 (attack) |
| 17.200000 (attack) | 19.000000 |
| 17.300000 (attack) | 20.000000 |
| 17.400000 (attack) | |
| 18.000000 | |
| 19.000000 | |
| 20.000000 | |

Figure 10.1: Data for additional evaluations

**Difficulty of parameter adjustment**

The results in Section 10.1.4 suggest that parameter adjustment is difficult for cycle detection, delayed-decision cycle detection, and Waszecki's method. These methods need to adjust their parameters in relation to the maximum deviation. The maximum deviation is due to collision delays in many cases. However, in some cases, the maximum early arrival and delay are due to deviations in the message itself. Collision delays can be predicted using a real-time scheduling method like in [63],[64]. On the other hand, it is difficult to predict the maximum early arrival and delay of the message itself. One solution is to use a long-term communication log for the prediction. It is thought that using a long-term log in a preliminary investigation makes for a highly accurate detection. However, there is no guarantee that the maximum early arrival or delay of the cycle reception will occur in the preliminary investigation log, and an incorrect prediction may lead to

false positives and negatives. In the evaluation using Waszecki's method, we derived the worst-case jitter from the log file. Therefore, the jitter was specialized to these data, and this led to very accurate detections. On the other hand, whereas the derivation of the worst-case jitter is generally difficult, the parameter adjustment of ADCS is very easy and does not require a sensitive parameter adjustment.

ADCS does not require any consideration of the deviation from the cycle reception timing; it only needs the number of early arrivals. This seems to be much easier than the parameter adjustments of the existing methods. Overall, ADCS has the best properties among the methods discussed here. Note that, as explained in Section 7.3.3, we should emphasize that the judged time is not exactly the same as the attack injection time, because all the methods described in this dissertation detect attacks at times that may not be exactly the same as the attack injection time.

ADCS and the existing methods are techniques that observe the message cycle. These techniques cannot detect an attack in which an attacker cancels normal messages and injects attack messages at the same timings. They also cannot detect message eavesdropping. Such attacks can be conducted by using remodeled CAN devices or tampered ECU. We discuss those attacks in Section 12.1.1, which are difficult to be conducted. Therefore, those attacks are not within in the main scope of this dissertation.

### 10.1.6 Evaluation of method 1 using CAN data from many vehicles

The above evaluations indicate that our method is more accurate than the existing methods. In this Section, we evaluate the detection accuracy of our method using many CAN data. These CAN data were collected from three actual vehicles (different from the one used in the evaluation above). We selected two messages from each vehicle (6 messages in total) and evaluated the number of detections for each message when attackers injected attacks in the manner described in Section 10.1.2. The results are shown in Table 10.4.

The number of attack messages and detected messages were the same for almost all messages. However, misdetections occurred in the case of message 3-2 of vehicles 3. The reason for these misdetections relates to the following property of message 3-2.

- Some intervals of consecutive reception times are large compared with other messages. On the other hand, delays caused by them are not recovered.

The reception intervals are shown in Figure 10.2, and 10.3. These figures indicate the reception intervals that are sorted from short one to long one. the reception intervals are sorted from short intervals to long intervals. Here, the reception intervals of message 3-2 are larger than those of message 3-1. In the case of message 3-2, the large intervals cause the received messages to deviate

from the expected period, namely, delays are caused. On the other hand, the delays are not recovered to normal intervals. We think this is the reason for the misdetections. Furthermore, we think this message is not perfectly periodic or quasi-periodic. On the other hand, the reception intervals of many messages are similar to those of message 3-1. Therefore, if the detectors analyze the reception intervals before using our method, our method will not cause such misdetections.

Table 10.4: Evaluation of proposal 1 using many actual logs from three vehicles

| Attack Pattern in Table 10.2 | Number of total attack messages | Number of messages detected as anomalous | | | | | |
|---|---|---|---|---|---|---|---|
| | | Vehicle 1 | | Vehicle 2 | | Vehicle 3 | |
| | | message 1-1 | message 1-2 | message 2-1 | message 2-2 | message 3-1 | message 3-2 |
| Pat.1 | 10000 | 10000 | 10000 | 10000 | 10000 | 10000 | 9719 |
| Pat.2 | 1000000 | 1000000 | 1000000 | 1000000 | 1000000 | 1000000 | 997291 |
| Pat.3 | 1000000 | 1000000 | 1000000 | 1000000 | 1000000 | 1000000 | 999473 |

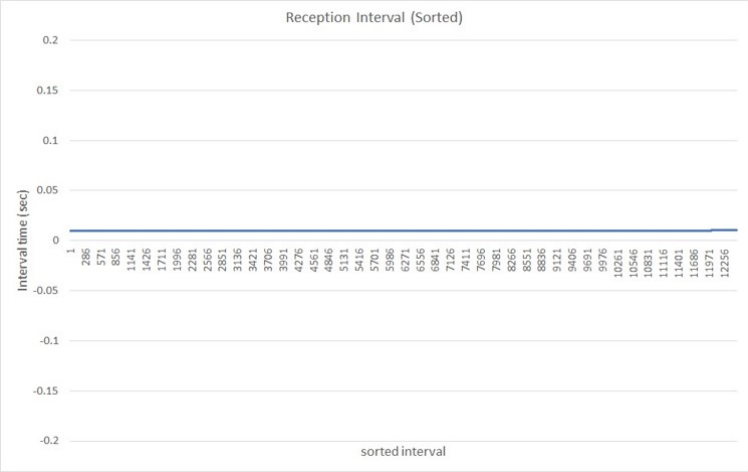


Figure 10.2: Message reception intervals of message 3-1 (Typical example)

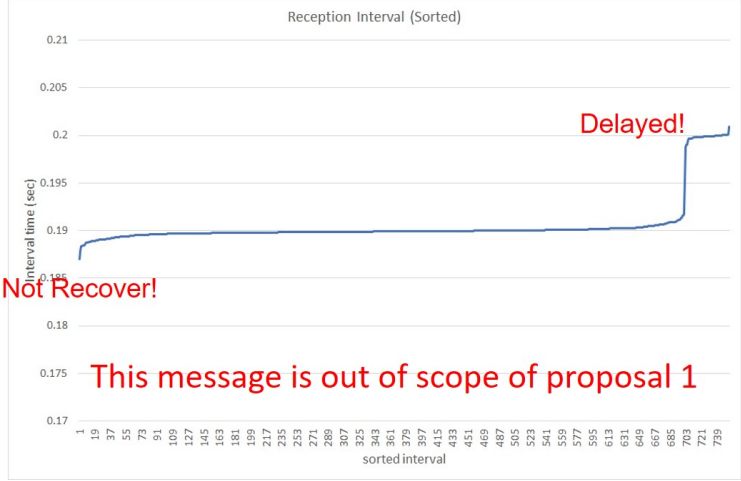


Figure 10.3: Message reception intervals of message 3-2

## 10.2 Evaluation of method 2

### 10.2.1 Evaluation method using CAN data from actual vehicles

We evaluated the estimation accuracy of our method and the existing methods described in Section 8.1.1 using CAN data from actual vehicles. For confirmation, we compared their accuracies with the results of a heuristic analysis. First, we collected CAN logs from two actual vehicles. The logs contained over 37 thousand reception of messages in over 300 seconds. The total number of the messages is over 100. Next, we chose 9 of the messages at random as the targets of the evaluation.

We used our method and the existing methods to estimate the field-data formats in the CAN payload for the chosen messages. Then, we compared the estimated results with the results of the heuristic analysis. Note that our method produced its estimation within several seconds on a generic PC.

We evaluated the format estimation accuracies of our method, Markovitz's method, and Kishikawa's method as follows.

$$Accuracy : \frac{Number\_of\_correct\_bits\_by\_estimation}{Message\_length} \tag{10.1}$$

We calculated the accuracy for each message with the same CAN-ID. After that, we averaged these accuracies (total accuracy).

Stone's method can only estimate data boundaries in CAN messages. We

used the tool on their web site [73] but found not estimate boundaries of the data from one of the vehicles. Therefore, we independently implemented their method for that vehicle. We evaluated the true positive ratio (TPR) and false positive ratio (FPR) of the boundary estimations of all methods, including Stone's method. These ratios were derived in the following manner.

$$TPR : \frac{Number\_of\_correct\_boundaries\_by\_estimation}{Number\_of\_true\_boundaries\_in\_message} \tag{10.2}$$

$$FPR : \frac{Number\_of\_incorrect\_boundaries\_by\_estimation}{Message\_length\_excluding\_true\_boundaries} \tag{10.3}$$

We calculated TPR and FPR for each message. Then, we averaged the TPRs and the FPRs (total TPR and FPR).

### 10.2.2 Results of format estimation

Some of the estimations are shown in Figure 10.4. The results of the evaluation (Table 10.5) indicate that our method estimated the format with the highest accuracy. It also estimated the boundaries of the data with a high TPR and low FPR.

### 10.2.3 Discussion

By using our method, an investigator can estimate many bits in the CAN payload correctly. Therefore, our method is helpful for determining rules for rules-based detection. In the case of a machine-learning-based method, the detection accuracy

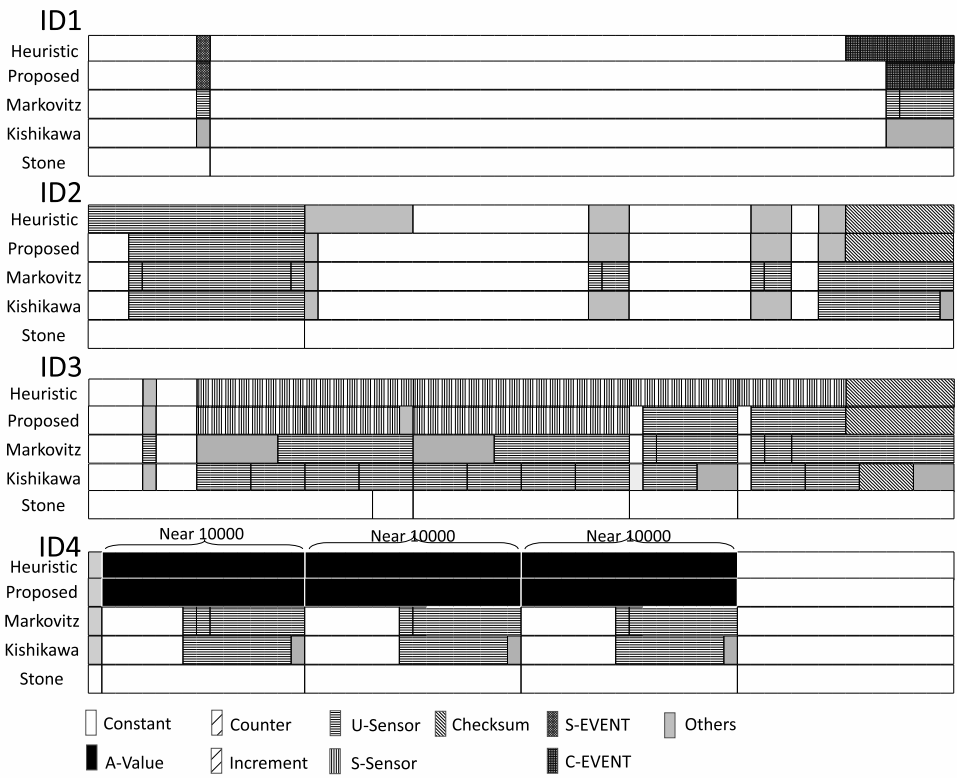


Figure 10.4: Example estimations from CAN-field data in actual vehicles

(Stone's method only estimates data boundaries.)

when using our method is expected to be better than when not using our method. We think that the reasons for the miss-estimations are as follows.

- The results of the heuristic analysis may include miss-estimations. However, we cannot confirm this because we don't know the correct answers.

- There are too few collected CAN logs. The lower bits of a field data are changed frequently in many cases if the field data has possibility of its value is changed. Among such field data, the higher bits of it may be changed by long-term observation, namely, large amount data. However, insufficient data may lead to a situation where changes of lower bits of the data don't cause carries to higher bits, and they are not changed. This carry-bit problem affects the Constant, Counter, Increment, U-Sensor, S-Sensor, and C-Event types. It can be solved by using more logs.

Table 10.5: estimation results (%)

| | Markovitz | Kishikawa | Stone | Proposed |
|---|---|---|---|---|
| Format estimation accuracy | 52.08 | 55.03 | - | 88.72 |
| Boundary estimation TPR | 81.56 | 81.56 | 49.85 | 89.35 |
| Boundary estimation FPR | 12.48 | 9.06 | 1.56 | 3.28 |

### 10.2.4 Evaluation method using artificially created data

In Section 10.2.1, we evaluated format estimation accuracy using CAN data from actual vehicles. However, since we don't know the correct format of data from actual vehicles, we could not confirm the strict correctness of results of estimation methods. In order to confirm the strict correctness, we evaluate accuracy of format estimation using data that we know the correct one. More concretely, We generate CAN data by combining data whose type is known. The followings is the generation method.

1. Make partial payload-data.

    (a) Select a message type of partial payload-data from ten types of field-data described in Section 8.2.2.

    (b) Select a length of the partial payload-data from four lengths (1 byte, 2 bytes, 3 bytes, 4 bytes).

    (c) According to the selected type and length, make a partial payload-data. One partial payload-data includes ten thousand time-series data that behaves according to the selected type.

    (d) Make all partial payload-data of all combinations of type and length. Totally, $10 \times 4 = 40$ partial payload-data and each partial payload-data include ten thousand time-series data are generated.

2. Make CAN data.

   (a) Select several partial payload-data from the made partial payload-data to generate CAN data whose length of payload is four bytes.

   (b) Generate a CAN data from the selected partial payload-data by concatinating them. Its CAN-ID is determined arbitrarily.

   (c) Generate all CAN data from all the combinations that can make 4-byte CAN data. Totally, 13300 CAN data are generated.

The estimation accuracy using the above data is described in Table 10.6.

Table 10.6: Estimation accuracy using artificially created data (%)

| | method 2 |
|---|---|
| Format estimation accuracy | 44.7 |
| Boundary estimation TPR | 57.9 |
| Boundary estimation FPR | 3.4 |

**Discussion**

The results of Section 10.2.4 seems that the accuracy is not so high. We think one important factor is that behavior of generated CAN data and that of actual vehicles may not be similar . For example, we think that the sensor messages and the event messages are incompatible. More strictly, we think that the sensor data

are sent periodically for confirmation of aliveness of transmission ECU. However, the event data are not sent periodically when an event occurs. Therefore, data of event types (S-event and C-event) and data of sensor types (S-sensor, U-sensor, and A-value) are not included in one CAN data. In order to confirm it, we evaluate the estimation accuracy excludes CAN data that includes both data of event types and sensor types. The accuracy is described in Table 10.7. It shows that the accuracy is improved. There could be similar problems with other types, and we think that the estimation accuracies for actual vehicles are higher than artificial one.

Table 10.7: Estimation accuracy excluding CAN data that includes both data of event types and sensor types (%)

| | method 2 |
|---|---|
| Format estimation accuracy | 53.6 |
| Boundary estimation TPR | 76.3 |
| Boundary estimation FPR | 4.5 |

### 10.2.5 Detection example using method 2

The results of the format estimation evaluation indicated that our method has the highest accuracy among the existing methods in Section 8.1.1. In this section, we utilize the results of our method for machine-learning-based detection.

### Preparation

We chose one CAN-ID as the detection target. The payload of this message was 64 bits. The estimation results for this message were “1-bit others”-“15-bit A-value”-“16-bit A-value”-“16-bit A-value”-“16-bit constant”. We prepared four formats, including this result (see Table 10.8).

First, we collected 45001 CAN payloads from the chosen messages. After that, we chose one prepared format and divided up all the CAN payloads according to the chosen format. In total, we prepared four datasets correspond to the formats. To detect anomalous messages, we trained machine learning model on each dataset in an unsupervised manner by using One-Class SVM with scikit-learn. We used the best hyper parameters for each training as much as possible. The parameters are listed in Table 10.9.

To consider relations between time sequences, we encapsulated ten messages into 1 input in the training and testing phases. Moreover, we trained the model using not only the divided-up payloads but also the differential of the time stamps.

Table 10.8: Prepared formats (division rules)

| | | |
|---|---|---|
| Format 1 | 1bit-1bit-,...,-1bit | (64×1bit-field-data) |
| Format 2 | 8bit-8bit-8bit-8bit-8bit-8bit-8bit-8bit | (8×1byte-field-data) |
| Format 3 | 1bit-15bit-16bit-16bit-16bit | (result of our estimation) |
| Format 4 | 32bit-32bit | (2×32bit-field-data) |

Table 10.9: Hyper parameters (in scikit-learn)

| Parameter | Set Value | Meaning |
|---|---|---|
| Kernel | rbf | Kernel type |
| Degree | Default | Degree of polynomial kernel function |
| Gamma | Auto | Kernel coefficient |
| Coef0 | Default | Independent term in kernel function |
| tol | Default | Tolerance for stopping criterion |
| nu | $10^{-1} - 10^{-10}$ | Bounds for SVM |
| shrinking | Default | Whether to use the shrinking heuristic |
| cache_size | Default | size of the kernel cache |
| max_iter | $-1$ | Hard limit on iterations within solver |

Then, we tested the trained models on 10919 test data including 100 anomalous messages. The anomalous messages were produced by hand. The test data were divided up in accordance with the prepared formats.

**Results**

We evaluated the detection accuracy, precision, recall, specificity, and F-measure. The results are shown in Figure 10.5. The accuracy and specificity are similar among the four formats. Our method had the best precision, recall, and F-measure for format 3, the best among the four formats. The times taken by the machine-

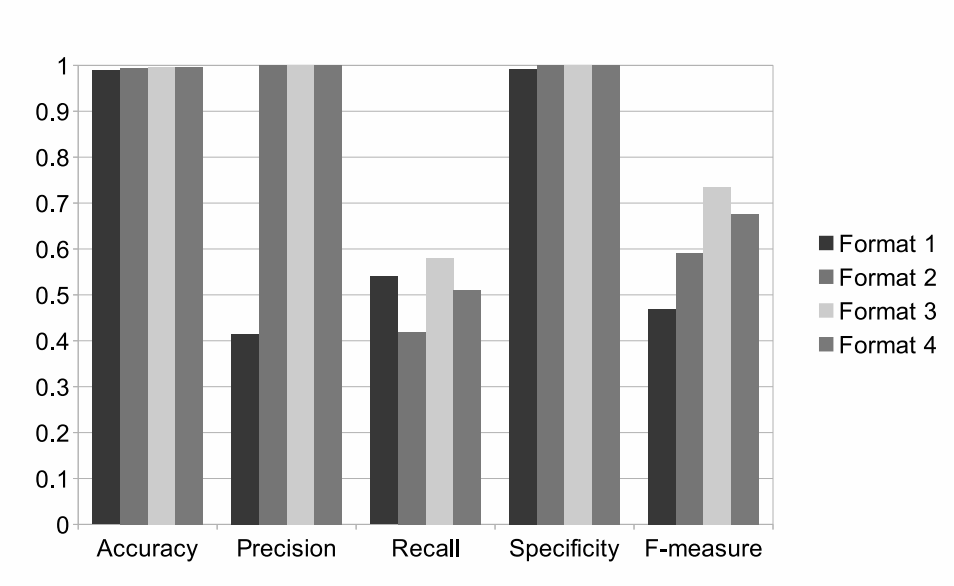


Figure 10.5: Detection accuracy

learning processes are shown in Table 10.10. Note that we executed all of the machine-learning processes on a CPU, not a GPU. The specifications of the computer are listed in Table 10.11.

**Discussion**

The results of this experiment show that recalls are not so high. We think that the reason for the low recalls is that the anomalous messages were prepared to be as hard to detect as possible even if a human looked for them.

Before conducting this experiment, we had considered that the shortest divi-

Table 10.10: Time required for machine-learning processes

| | Training (Sec) | Testing (Sec) |
|---|---|---|
| Format 1 | 7.25 | 1.30 |
| Format 2 | 0.64 | 0.31 |
| Format 3 | 0.043 | 0.032 |
| Format 4 | 0.048 | 0.031 |

Table 10.11: Computer specifications

| | |
|---|---|
| CPU | Intel Core i7-8700 3.2GHz |
| Memory | 16GB |
| OS | Ubuntu 18.04.2 |

sion (format 1) would give the highest detection performance. Because format 1 has many more boundaries than the other formats, we thought it would have many useful feature values for the detection. However, the results of the detection experiment indicated that format 1 was the least accurate among the four formats. One reason is that the shortest division has many more field data (feature values) compared with the other formats. In general, machine learning requires many training data when the dataset has many feature values. Therefore, we think that the number of required training and test data for this format is more than that for other formats. To confirm this for format 1, we prepared ten times more data as in the previous experiment. The results showed that the detection accuracy was

not much different from the previous result. Therefore, we think that main factor affecting detection accuracy is not the amount of data but rather the division rule (format).

## 10.3 Evaluation of method 3

We conducted a computer experiment on the data relation analysis method.

### 10.3.1 Preparation of data

We collected CAN data from an actual car and ran our method on the CAN data. We selected related data whose number of changes, or matches a specific value were the same as well as data whose deviation-characteristic from its period was the same. We set the targets whose number of changes or matches a specific value to 2 at minimum and 64 at maximum.

Table 10.12: Results of the experiment

| | |
|---|---|
| Log length | about 315 seconds |
| The variation of IDs | 122 patterns |
| Total messages | 376873 |
| Extracted relations | 582 |

Table 10.13: Example of a relation (number of changes/matches is 17)

| | |
|---|---|
| ID-$X$ | (2, 8, 166),(8, 8, 128),(59, 5, -1) |
| ID-$Y$ | (1, 8, 85),(8, 8, 128),(59, 5, -1) |

### 10.3.2 Results

As shown in Table 10.12, we found 582 relations from the CAN data. An example of a found relation is shown in Table 10.13: the triplet $(a, b, c)$ indicates $(bitposition, datalength, datatype)$, where $datatype$ means that

$datalength$-bit data is at position $bitposition$, and its value becomes $datatype$ when the message intervals are deviated. When $datatype$ is "$-1$", the data changes to another data when the message intervals are deviated. In Table 10.13, for messages with ID-$X$ and ID-$Y$, the 8-bit length data at bit position 8 become 128 (0x80) when the message intervals are deviated. Figure 10.6 shows part of the CAN data. From this log, we can confirm that the characteristics of data $(8, 8, 128)$ in the message with ID-$X$ are the same as those of the message with ID-$Y$. Table 10.14 shows the number of relations for each number of changes/matches.

| ID | Time | Data(HEX) | ID | Time | Data(HEX) |
|---|---|---|---|---|---|
| X | 57.043154 | 29 00 00 00 00 00 28 4C | Y | 57.054185 | 2A 00 00 00 00 00 28 4C |
| X | **57.943272** | 29 **80** 00 00 00 00 28 4D | Y | **57.954177** | 2A **80** 00 00 00 00 28 4D |
| X | 67.942846 | 29 00 00 00 00 00 28 4D | Y | 67.953193 | 2A 00 00 00 00 00 28 4D |
| X | 77.942821 | 29 00 00 00 00 00 28 4D | Y | 77.954915 | 2A 00 00 00 00 00 28 4D |
| X | **79.122813** | 29 **80** 00 00 00 00 28 4E | Y | **79.133920** | 2A **80** 00 00 00 00 28 4E |
| X | 89.122655 | 29 00 00 00 00 00 28 4E | Y | 89.132772 | 2A 00 00 00 00 00 28 4E |
| X | **92.002538** | 29 **80** 00 00 00 00 28 4F | Y | **92.012815** | 2A **80** 00 00 00 00 28 4F |
| X | **100.682696** | 29 **80** 00 00 00 00 28 50 | Y | **100.694782** | 2A **80** 00 00 00 00 28 50 |

Figure 10.6: Part of log on ID-*X* and ID-*Y*

Table 10.14: Found relations

| Num. (∗1) | Rel. (∗2) | Num. | Rel. | Num. | Rel. | Num. | Rel. | Num. | Rel. | Num. | Rel. | Num. | Rel. | Num. | Rel. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 315 | 10 | 5 | 18 | 1 | 26 | 0 | 34 | 0 | 42 | 0 | 50 | 0 | 58 | 0 |
| 3 | 126 | 11 | 6 | 19 | 0 | 27 | 0 | 35 | 0 | 43 | 0 | 51 | 0 | 59 | 0 |
| 4 | 28 | 12 | 5 | 20 | 4 | 28 | 1 | 36 | 0 | 44 | 0 | 52 | 0 | 60 | 0 |
| 5 | 12 | 13 | 3 | 21 | 3 | 29 | 1 | 37 | 0 | 45 | 0 | 53 | 0 | 61 | 0 |
| 6 | 33 | 14 | 0 | 22 | 0 | 30 | 0 | 38 | 0 | 46 | 0 | 54 | 0 | 62 | 0 |
| 7 | 14 | 15 | 4 | 23 | 1 | 31 | 0 | 39 | 0 | 47 | 0 | 55 | 0 | 63 | 3 |
| 8 | 9 | 16 | 1 | 24 | 1 | 32 | 0 | 40 | 0 | 48 | 0 | 56 | 0 | 64 | 0 |
| 9 | 3 | 17 | 1 | 25 | 1 | 33 | 0 | 41 | 1 | 49 | 0 | 57 | 0 | | |

∗1: Number of changes/matches, ∗2: Found relations

### 10.3.3 Discussion

The results in Table 10.14 reveals many field data that have relations each other. We can also see that low numbers of changing/matching have many field data that have relations each other. In this subsection, we will discuss the reason for this result and what should be done. We think the reason is that when there are few changes in value or few matches in value to a specific value, there are few checks in step 3 in algorithm 2. Thus, even if the field data don't have relations between sets $S_{i,j1}$ and $S_{i,j2}$, the algorithm judges that $S_{i,j1}$ and $S_{i,j2}$ have the same characteristics because of the small number of changes/matches. Therefore, in the detection phase, relations that have few changes or matches a specific value should be excluded. To exclude them, the appropriate thresholds may be needed.

## 10.4 Validity of the evaluations

### 10.4.1 Evaluation of method 1

The target messages (perfectly periodic and quasi-periodic) were evaluated. Furthermore, we collected a lot of CAN data from actual vehicles for additional evaluations. We think the amount of data was adequate for the evaluation using the PC simulation, although the operations in the simulation may have been different from those of actual ECUs. However, we think that the complexity of our method

is lower than that of typical existing methods and thus that it can detect attacks on actual ECUs. We think the results are valid because they are similar to the theoretically expected results.

### 10.4.2 Evaluation of method 2

We used randomly selected messages, including A-value, S-Event, and C-Event. Therefore, we think the evaluation is valid for these types. We confirmed that our method can estimate boundaries accurately. Therefore, we think that the results of the anomaly detection experiment showing that method 2 has the highest accuracy of the methods tested are valid.

### 10.4.3 Evaluation of method 3

We found many relations for constructing anomaly detections and confirmed that one of the relations can be used to detect anomalies. Hence, we think the results are valid. However, because method 3 finds so many relations, it may be necessary to develop a means of automatically selecting them for use in a detector.

# Chapter 11

# Examples of Detection Using Our Methods

## 11.1 Implementations of rules-based detection

Section 6.4.1 describes method 1 as a detection method, while methods 2 and 3 are described as useful techniques for constructing detection methods. In this section, we introduce some examples of detection methods constructed using the messaging behaviors revealed by methods 2 and 3.

### 11.1.1 Focusing on the behavior of each type of field data

As described in Section 6.1.1, method 2 reveals the behavior of each type of field data. We found that analyzing of the normal behavior of the data in the CAN payload is very important for constructing accurate detection methods. For example, when the detector knows that the message period is 10 milliseconds and also knows that the data changes only slightly during one period (ex. steering angle information: the change in value is naturally limited by the driver's steering ability), if this value changes by a large amount, the situation can be judged to be abnormal. In the same manner, detection procedures can be constructed for each type of behavior. Therefore, knowing the type and boundaries of the field data in the CAN payload is very important.

Format information is not published, though it may be known to car manufacturers. In particular, manufacturers may know part of the format related to the area of the field data in the CAN payload, but not the behavior of the field data in detail. If they don't know the behavior of the field data, they can construct detection methods by using the results of method 2. If they know the behavior of the field data, they can confirm whether their information is correct or not by comparing the results of method 2. Moreover, car manufactureres may not know the format of messages sent from equipments of third party maker that may be installed after released. Our method can be used to such messages. For these reasons, we

constructed the format estimation method shown in Chapter 8.

### 11.1.2 Focusing on relations between messages

Method 3 reveals relations between multiple messages and in the experiments, it found that some field data relate to each other. Figure 11.1 is an example of a CAN log that shows such a relation.

(a)CAN log

| ID | Time | Data(HEX) |
|---|---|---|
| 100 | 0.500001 | 00 01 12 13 |
| 200 | 0.600005 | 00 F2 E3 C5 |
| 100 | 0.799997 | 00 01 12 13 |
| 200 | 0.900002 | 00 F2 E3 C5 |
| 100 | 1.100004 | 00 01 12 13 |
| 200 | 1.200003 | 00 F2 E3 C5 |
| 100 | 1.200503 | **80** 01 12 13 |
| 200 | 1.300498 | 00 F2 E3 **45** |
| 100 | 1.500505 | 00 01 12 13 |
| 200 | 1.600502 | 00 F2 E3 45 |

(b)ID=0x100

| ID | Time | Data(HEX) |
|---|---|---|
| 100 | 0.500001 | 00 01 12 13 |
| 100 | 0.799997 | 00 01 12 13 |
| 100 | 1.100004 | 00 01 12 13 |
| 100 | 1.200503 | **80** 01 12 13 |
| 100 | 1.500505 | 00 01 12 13 |

(c)ID=0x200

| ID | Time | Data(HEX) |
|---|---|---|
| 200 | 0.600005 | 00 F2 E3 C5 |
| 200 | 0.900002 | 00 F2 E3 C5 |
| 200 | 1.200003 | 00 F2 E3 C5 |
| 200 | 1.300498 | 00 F2 E3 **45** |
| 200 | 1.600502 | 00 F2 E3 45 |

Figure 11.1: Example of a relation

Figure 11.1-(b) and Figure 11.1-(c) show the logs for each ID. In this example, for ID=0x100 and 0x200, messages are received at intervals of 0.3 second when no event occurs. On the other hand, when an event occurs, the message interval is not about 0.3 seconds, and the messages with ID=0x200 are received about 0.1 seconds after messages with ID=0x100. Moreover, byte 1 of the messages with

ID=0x100 and byte 4 of messages with ID=0x200 seem to have same behavior. Namely, we have found the following property.

**Property** The number of specific values ($0x80$) in the 1st byte of messages with ID=0x100 is the same as the number of changes in value in the 4th byte of messages with ID=0x200.

From this finding, we can construct the following detection method:

1. When message 0x100 is received, the detectors judge whether an event has occurred by checking whether the 1st byte takes on a specified value. If the 1st byte is not equal to the specified value, the message reception interval should be checked, and if it is not equal to the message period, the situation is judged to be anomalous. If the 1st byte is equal to the specified value, wait for reception of message 0x200.

2. When message 0x200 is received, detectors judge whether an event has occurred by checking for changes of the value of the 4th byte. If the value of the 4th byte does not change, message reception interval should be checked, and if it is not equal to the message period, the situation is judged to be anomalous. If the value of the 4th byte changes, the interval of reception times of messages 0x100 and 0x200 is checked, and if it is not equal to about 0.1 second, the situation is judged to be anomalous.

When the detector tries to detect attacks using this method, the detector must know the position of the specific/changed value, and the number of the specific/changed values before attacks are injected. Therefore, finding relations is important. Our finding method of relations are shown in Chapter 9. The detector in this case must know the position of the specified and changed value.

### 11.1.3 Estimation of detection accuracy of rule-based detection

**Expected accuracy of detection method based on proposal 2**

The expected detection accuracy of rule-based detection using proposal 2 $Acc_2$ is estimated by the following manner when we can estimate the correct format.

$$\begin{aligned} Acc_2 &= P(Detect\_anomaly\_on\_fielddata\_1) \\ &\quad \cup P(Detect\_anomaly\_on\_fielddata\_2) \\ &\quad \cup ... \cup P(Detect\_anomaly\_on\_fielddata\_n) \end{aligned}$$

On the above equation, $n$ is the number of field-data in the message which is the detection target. $P(Detect\_anomaly\_on\_fielddata\_i)$ is the detection probability on the $i$-th field-data. The accuracy can be calculated for each message. The total accuracy $TAcc$ can be calculated by using all accuracies that is calculated by each target, and transmission-probability of each message may be considered to calculate $TAcc$.

The detection accuracy on each field-data depend on the field type. The detection accuracy for each field type is summarized at Table 11.1.

The reasons of the accuracies are the follows.

- A-value

  Data of this type follow the normal distribution of $N(Appropriate\_value\ (Avalue), \sigma_i^2)$, and $\sigma_i$ depends on each appropriate value. For anomaly detection, we set permissible boundaries $w$ around the value of the previous received message. Namely, we judge as normal data when the previous value of data $V_p - w \leq$ received value $\leq V_p + w$. On clever attackers, we assume that they know the distribution. So, the detection accuracy is equal to the value in Table 11.1. On the other hand, we assume that casual attackers don't know the distribution. They inject messages with random value. So, the detection accuracy is equal to the value in Table 11.1.

- S-sensor

  Data of this type follow the normal distribution of $N(0, \sigma_i^2)$, and $\sigma_i$ depends on each sensor value. For anomaly detection, we set permissible boundaries $w$ around the value of the previous received message. Namely, we judge as normal data when the previous value of data $V_p - w \leq$ received value $\leq V_p + w$. On clever attackers, we assume that they know the distribution. So, the detection accuracy is equal to the value in Table 11.1. On the other

Table 11.1: Detection accuracy of each field-data

| Field-data type | Accuracy for clever attackers | Accuracy for casual attackers |
|---|---|---|
| A-value | $\int_{-\infty}^{+\infty}\int_{y-w}^{y+w} dxdyf(x)f(y)$<br>$f(x) = N(Avalue, \sigma_2^2)$<br>$f(y) = N(Avalue, \sigma_2^2)$ | $\int_{-\infty}^{+\infty}\int_{y-w}^{y+w} dxdyf(x)f(y)$<br>$f(x) = N(Avalue, \sigma_2^2)$<br>$f(y) = \frac{1}{2^{fieldlen}}$ |
| S-sensor | $\int_{-\infty}^{+\infty}\int_{y-w}^{y+w} dxdyf(x)f(y)$<br>$f(x) = N(0, \sigma^2)$<br>$f(y) = N(0, \sigma^2)$ | $\int_{-\infty}^{+\infty}\int_{y-w}^{y+w} dxdyf(x)f(y)$<br>$f(x) = N(0, \sigma^2)$<br>$f(y) = \frac{1}{2^{fieldlen}}$ |
| U-sensor | $\int_{-\infty}^{+\infty}\int_{y-w}^{y+w} dxdyf(x)f(y)$<br>$f(x) = N(\mu, \sigma^2)$<br>$f(y) = N(\mu, \sigma^2)$ | $\int_{-\infty}^{+\infty}\int_{y-w}^{y+w} dxdyf(x)f(y)$<br>$f(x) = N(\mu, \sigma^2)$<br>$f(y) = \frac{1}{2^{fieldlen}}$ |
| Counter | 1 | 1 |
| Increment | almost $\frac{1}{m}$ | $\frac{2^{fieldlen}-m}{2^{fieldlen}}$ |
| Checksum | 0 | $\frac{2^{fieldlen}-1}{2^{fieldlen}}$ |
| C-Event | 0 | almost 1 |
| S-Event | probability of occurrence<br>of no event | almost 1 |
| Constant | 0 | $\frac{2^{fieldlen}-1}{2^{fieldlen}}$ |
| Others | 0 | 0 |

hand, we assume that casual attackers don't know the distribution. They inject messages with random value. So, the detection accuracy is equal to the value in Table 11.1.

- U-sensor

  Data of this type follow the normal distribution of $N(\mu, \sigma_i^2)$ with $x$-Axis $\geq 0$, and $\sigma_i$ depends on each sensor value. For anomaly detection, we set permissible boundaries $w$ around the value of the previous received message. Namely, we judge as normal data when the previous value of data $V_p - w \leq$ received value $\leq V_p + w$. On clever attackers, we assume that they know the distribution. So, the detection accuracy is equal to the value in Table 11.1. On the other hand, we assume that casual attackers don't know the distribution. They inject messages with random value. So, the detection accuracy is equal to the value in Table 11.1.

- Counter

  Data of this type is incremented 1 for each reception. On clever attackers, we assume that they know this behavior. However, when they inject attack message with value $V_p + 1$, we know the attack because the next legitimate message has the value $V_p + 1$. Therefore, messages with the value $V_p + 1$ are received twice. So, we can aware of the attack with probability 1. On the casual attackers, we can also aware the attack with probability 1.

- Increment

  Data of this type is incremented a few for each reception. On clever attackers, we assume that they know this behavior. When they inject attacks with value $V_p + 1$, we cannot detect attack in many cases. When average increment amount is $m$, the detection accuracy is almost $\frac{1}{m}$. On the casual attackers, their attack message is judged as normal when random value is lower than $m$, and higher than $V_p$. So, the detection accuracy is equal to the value in Table 11.1.

- Checksum

  Data of this type is calculated by the defined manner. On clever attackers, we assume that they know the manner. So, the detection accuracy is 0. On the casual attackers, their attack message is judged as normal when random value is equal to the correct value. So, the detection accuracy is equal to the value in Table 11.1.

- C-event

  Data of this type is periodically sent in the normal situations, and not periodically sent in the event situations. So, when attack messages are injected in the normal situations, we can detect attacks easily with reception counter like proposal 1. On clever attackers, we assume that they know this behavior. So, they focus on the event situations. On this type, data-value is

changed to other values when an event is occurred. Therefore, they inject attacks with another value of normal situations. In this case, we cannot detect attacks. On the other hand, messages from casual attackers can be detected easily.

- S-event

  Data of this type is periodically sent in the normal situations, and not periodically sent in the event situations. So, when attack messages are injected in the normal situations, we can detect attacks easily with reception counter like proposal 1. On clever attackers, we assume that they know this behavior. So, they focus on the event situations. On this type, data-value matches a specific value when an event is occurred. Therefore, they inject attacks with the specific value of event situations. However, we can detect their attacks by monitoring message reception intervals. If the value has a specific value, we think that the situation is an event situation. However, the reception interval after injected message is not equal to the normal periodicity. So, we can detect them exclude the situation is just equal to the true event situations. On the other hand, messages from casual attackers can be detected easily.

- Constant

  Data of this type is constant. On clever attackers, we assume that they know

the constant value. So, the detection accuracy is 0. On the casual attackers, their attack message is judged as normal when random value is equal to the constant value. So, the detection accuracy is equal to the value in Table 11.1.

- Others

  Data of this type is undefined. So, we cannot construct detection method. So, detection accuracy is 0.

**Accuracy of detection method based on proposal 3**

Detection method based on proposal 3 work accurately for the event messages when we can use appropriate relations. In the previous paragraph, we discussed the expected detection accuracy for single event message. Based on the discussion, we can detect attacks with almost probability 1 for casual attackers. For clever attackers, we assume that they know the event behavior. By the previous discussion, we can detect attacks when data type is S-event with probability of occurrence of no event. Therefore, the probability of detection using found relations is described in Table 11.2. This table shows detection accuracy for related two messages (message 1 and 2). When message 1 and 2 are both S-event, the accuracy is equal to the probability of occurrence of no event because two messages have same behavior (depend each other).

Table 11.2: Detection accuracy using proposal 3 for clever attackers

| | | message 1 | |
|---|---|---|---|
| | | S-event | C-event |
| message 2 | S-event | Probability of occurrence of no event | Probability of occurrence of no event |
| | C-event | Probability of occurrence of noevent | 0 |

## 11.2 Example of machine-learning-based detection

One method of detection is as follows.

1. (Preparation) Divide up the CAN logs into field data with boundaries.

2. (Training phase) Train a machine learning model using divided field data.

3. (Prediction phase) Input divided-up received-CAN messages to the trained machine learning model, which outputs predicted anomalies.

Machine-learning-based detection requires many CAN logs for training. In the training phase, the model is automatically trained using divided field data by using machine learning algorithms. However, it is difficult to find correct boundaries to divide payloads of CAN log because there are many pattern combinations for each boundary. We consider that the detection accuracy when using a model trained on field data without correct boundaries is lower than that of one trained

on field data with correct boundaries. Therefore, format estimation, especially boundary estimation, is important for accurate detection, and method 2 can reveal the boundaries of the field data.

# Chapter 12

# Discussion

## 12.1 Attacks against proposed methods

We think that our methods will work even if the attackers know them. In this section, we discuss the effectiveness of our methods.

### 12.1.1 Attacks against method 1

This technique is for perfectly periodic and quasi-periodic messages. If a vehicle satisfies the assumptions described in Section 7.3.1, a detection unit (gateway or detection specialized unit on in-vehicle network) can detect attacks even when only one attack message is injected in the in-vehicle network. Therefore, even if attackers know that this method is implemented in a vehicle, they cannot inject attack messages that can evade detection. The above discussion indicates that the

effectiveness of method 1 depends on whether a vehicle meets our assumptions described in Section 7.3.1. We discuss the appropriateness of the assumptions below.

**Appropriateness of our assumptions**

Our assumptions described in Section 7.3.1 are as follows.

1. All attack messages are injected from the external network through the external communication unit or the OBD-II port.
2. Attackers cannot hijack ECUs via in-vehicle networks equipped for the CAN protocol.
3. The externally connected ECUs are not related to the essential behavior of the car, such as a control system.
4. The message reception times may have delays or early arrivals.
5. All legitimately messages are sent and received.

Regarding assumption 1, all our methods are for indirect attacks where the attackers inject attacks from the outside a car. Thus, all messages from outside the car pass through external communication units, meaning that attack messages are injected from external networks via the external communication units. Therefore, assumption 1 is met.

Regarding assumption 2, the CAN payload is only 64 bits at maximum. In addition, its transmission speed is only 500 kbps in general operation, and networks become congested when many ECUs transmit messages. Therefore, only essential values (sensor values, checksum values, or counter values) are sent on in-vehicle networks using CAN. This means that operation codes are not sent on in-vehicle networks using CAN. This can be seen from the CAN format information published on the Internet [74], although it is for only one vehicle type. Therefore, hijacking ECUs via in-vehicle networks using CAN is almost impossible. Thus, assumption 2 is met.

Regarding assumption 3, the discussion in regard to assumption 2 means that hijacking is not conducted via in-vehicle networks using CAN. Vehicles typically have many ECUs (over 100), and the in-vehicle network allocates a role to each one. The vehicle is controlled by ECUs specialized for that purpose. Similarly, external connections are made through an external connection unit. This means that in almost all cases, the control ECUs are not exposed to the external network. The only exception is shown in Figure 6.7, where the self-driving unit are connected to other ECUs by TCP/IP. In this case, we have already discussed that, by themselves, our techniques can't detect attacks in these special cases. Excluding these cases, though, assumption 3 is met.

Assumptions 4 and 5 mean that our method can be applied to any type of in-vehicle network. Our investigations indicate that assumption 4 is true for over

60% of messages in in-vehicle networks using CAN. This means that over 60% of messages are perfectly periodic or quasi-periodic. Among them, we could not find any messages that didn't meet assumption 5.

Therefore, we consider our assumptions are appropriate for perfectly periodic and quasi-periodic messages. Methods 2 and 3 can be applied to the remaining message types. Moreover, they can be applied to perfectly periodic and quasi-periodic messages, too (see Section 6.3).

**Message replacement attack**

Method 1 works effectively in situations in which legitimate periodic messages are sent periodically and attack messages are sent on the same CAN bus. Therefore, we presume that attacks cannot be detected if the legitimate messages are stopped and attack messages are sent periodically in the legitimate period. Let us consider two ways of stopping legitimate messages:

- hijack a legitimate ECU that sends legitimate messages;
- overwrite legitimate messages by transmitting error frames.

The hijack method contradicts assumption 2, because, as pointed out in the above discussion, attackers cannot hijack a legitimate ECU that only connects to the CAN bus. The overwrite method is mentioned in [75]. To conduct message overwriting, attackers must hijack an ECU that is connected to the target

ECUs which are connected to the CAN bus and other ECUs with other protocols (i.e. TCP/IP). After that, the attackers would inject error frames in the CAN bus. However, error-frame injection is not supported by the general application interface (API) of the CAN controller. In [75], the authors pointed out that their method requires a physical layer implementation in order to achieve a sufficient real-time response. Therefore, the attacker must prepare a modified controller to inject error frames, or else tamper with the firmware of the hijacked ECU to modify the API. Because attackers must use the originally installed ECUs in indirect attacks, the former attack cannot be conducted. Moreover, conducting the latter attack is very difficult because the firmware cannot be tampered in a secure vehicle equipped with a tampering prevention mechanism like the secure boot described in Section 3.1. This means that a message replacement attack is very difficult to launch.

**Reinitialization attack**

In method 1, reinitialization processing of counters is needed to use this method long term. We found that method 1 miss-detects in very rare cases when attackers inject attack messages at the reinitialization timing. In such a situation, we recommend that the reinitialization timing be randomly set, as proposed in [60]. This technique makes it impossible for attackers to inject attack message at the reinitialization timings. This countermeasure dramatically reduces such rare cases

miss-detections.

### 12.1.2 Attacks against method 2

Method 2 is not a detection method, but rather a support for constructing detection methods.

#### Within threshold value attack

An example detection method is checking whether the behavior of the field data deviates from normal. In this method, when one of the sensor values exceeds a threshold value, the situation is judged to be an attack. Attackers may inject attacks with messages that have sensor values within the threshold value if attackers know this detection method is being used and they know the threshold value. We think that the detection method cannot detect attack messages in this situation. However, the threshold value is very difficult to learn. When the threshold value depends on other sensor values (e.g., the speed of the vehicle), it becomes even more difficult to learn. On the other hand, if attackers know the threshold values and inject attack messages within the range of the threshold values, all of the injected values will be within the range of the threshold values. If the injected values are around the values of legitimate messages, the target car moves almost correctly. On the other hand, if the injected values are far from the values of legitimate messages, the target car moves in a dangerous manner. Therefore, the

detection method should utilize the distance between messages in order to detect attacks when the distance is large.

**Payload encryption from car manufacturer**

If the car manufacturer does not want the format information to be revealed, the payload may be encrypted. Division of labor is progressing in the development of ECUs, and not all ECUs are developed by car manufacturers [76]. This makes key management of the encryption algorithm is very difficult. In particular, we think payload encryption causes the following problems.

- Simple encryption such as exclusive-or of the secret key and plaintext (payload) is breakable. When the format is known, exclusive-or of the ciphertext and partly known plaintext will easily derive the secret key. To avoid this problem, the car manufacturer can dramatically change the format of the payload. However, doing so would likely cause a huge increase in development costs.

- Secure encryption such as using stream ciphers faces a performance problem that limits its use to certain messages.

### 12.1.3 Attacks against method 3

Method 3 is not detection method; rather it supports the construction of detection methods. The attacks against method 2 are the same as those against method 3. We think that their solutions are also the same. Therefore, we will omit any discussion of them.

## 12.2 Advantages, disadvantages and limitations of our methods

Section 12.1 indicated that our methods work effectively even if attackers know they are being applied. This section discusses the advantages, disadvantages and limitations of our methods.

### 12.2.1 Proposal 1

#### Advantages

This method is for perfectly periodic and quasi-periodic messages. It can detect anomalous messages with almost no false positives or negatives under the assumptions described in Section 7.3.1. The assumptions are met in most situations discussed in Section 12.1.1.

#### Disadvantages

This method can only detect situations of attacks. It cannot distinguish attack messages from normal messages directly. However, most of the existing methods based on message periodicity cannot distinguish them either.

#### Limitations

This method doesn't detect anomalous event-based or non-periodic messages. Moreover, it doesn't work when the target message doesn't obey the assumptions. However, there are few such messages.

### 12.2.2 Proposal 2

#### Advantages

The detection methods constructed with this method are for all types of messages. Method 2 is more accurate than the existing estimation methods.

#### Disadvantages

Detection accuracy is lower than that of method 1 for perfectly periodic and quasi-periodic messages. Moreover, detection accuracy deteriorates when method 2 causes a miss-estimation.

#### Limitations

Our method is not accurate when there are many unknown messages types. On the other hand, it can be applied to general messages.

### 12.2.3 Proposal 3

#### Advantages

Our method can reveal relations between multiple messages that cannot be revealed by any of the existing methods.

#### Disadvantages

Our method reveals many relations between multiple messages. However, some of them cannot be used for detection. Currently, the judgement of whether relations are applicable to detection or not must be conducted heuristically.

#### Limitations

Not all of the applicable relations may found when there are many of them beyond the capability of humans to judge their applicability.

## 12.3 Relation between attack detection and vehicle control

The discussion in 6.2 indicates that the external network connection should be cut when attacks are detected. In addition, some part of the in-vehicle network should also be cut off from the control system of the in-vehicle network. This means an emergency shutout unit is needed. The detection ECUs (gateway or specialized ECUs) and the emergency shutout units should be connected through an exclusive line for emergency alerts. In addition, an emergency control unit may be needed. The emergency control unit controls the car safely by reducing its speed and stopping it on the road shoulder.

Using message authentication code (MAC) is another solution. The discussion in Section 4.2 points out that MAC cannot be used for all CAN messages. In other words, MAC works on only some messages. A method that detects only attacks on messages that use MAC is discussed in [77][78].

## 12.4 Ideal network structure

Our ideal network structure is illustrated in Figure 12.1. The detection method of proposal 1 and the detection methods constructed by proposal 2 and 3, is installed in the IDSs in the figure. As discussed in Section 12.3, the IDS sends emergency

messages over the exclusive line to the emergency shutout Units (ESs) when an anomaly message is detected. Then, the ESs cut network traffic when an emergency message sent from the IDSs is received via the exclusive line. After that, the emergency control unit controls the car safely by reducing its speed and stopping it on the road shoulder.

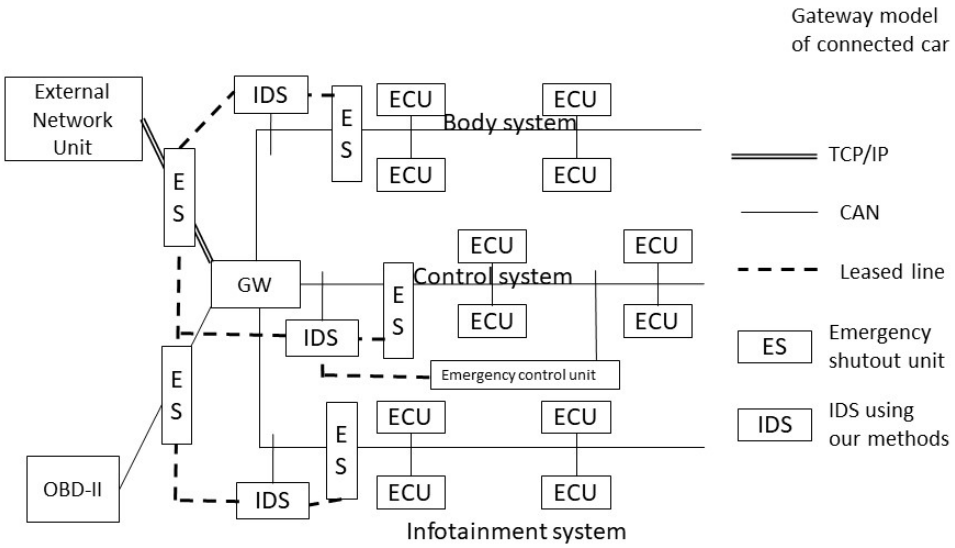


Figure 12.1: Ideal network structure

## 12.5 For self-driving cars

Our methods can be used on self-driving cars.

Method 1 can be used under the assumptions described above, which are met by self-driving cars.

We have presented a detection method constructed using threshold values determined by methods 2 and 3. We think that transmitted values of messages in self-driving cars are more estimable than those of manual-drive cars. The thresholds for self-driving cars more severe than those for manual-drive cars. Therefore, we think that the presented detection method should be especially effective for self-driving cars.

# Chapter 13

# Conclusion

In this dissertation, we explained cyberattacks against connected cars and self-driving cars. At first, we explained existing security strategies for IT environments. We considered security strategies for vehicles and decided to focus on detection technologies. We introduced the in-vehicle protocols and selected CAN as the target of our research. We described the current attack detection technologies for vehicles, including the detection methods, assumed networks, and requirements. After that, we proposed a new anomaly detection method, a new format estimation method, and a new data relation analysis method.

Our anomaly detection method is for perfectly periodic and quasi-periodic messages. Called "Anomaly Detection by Cumulative Sum ," this method can detect attacks with almost no false positives or negatives when large delays and early arrivals occur and the reception cycle of the periodic transmission message is bi-

ased. We evaluated the detection accuracies of anomaly detection by cumulative sum, cycle detection, delayed-decision cycle detection, and Waszecki's method and found that anomaly detection by cumulative sum and Waszecki's method had the highest detection accuracy among the four. However, Waszecki's method needs precise parameter adjustments for it to have high detection accuracy, while anomaly detection by cumulative sum requires little or no parameter adjustment. Therefore, we think that anomaly detection by cumulative sum is an excellent method from the viewpoint of detection accuracy and parameter adjustment.

Our format estimation method, called "Single Message Format Estimation," helps us to construct anomaly detection methods. This method can estimate ten field-data types in the CAN payload and the boundaries for each type of field data. We evaluated the estimation accuracy of the format and the data boundaries in a computer experiment. We found that our method can estimate the field-data format correctly about 88.7% in this time. It also can estimate field-data boundaries with 88.4% TPR and 3.3% FPR. We compared our method with the existing methods and found it was the most accurate among them. We also evaluated the detection accuracy by using a detection method with one-class SVM unsupervised learning using formats including those determined by our method and other payload division methods. As a result, the anomaly detection using the format estimated by our method had the highest accuracy among the four formats.

Our relation analysis method, called "Multiple Messages Relation Analysis,"

is also helpful for constructing anomaly detection methods. This method can find 582 data relations from pre-collected CAN logs. We confirmed its effectiveness in a computer experiment and found many relations that can be used for attack detection.

We conclude that our methods can detect cyberattacks against vehicles with high accuracy.

We explained the appropriateness of our evaluations by considering some viewpoints (i.e. data amount, the complexity, the results, etc.). We showed some examples of detection methods constructed by using our methods. We discussed the possible attacks against our methods and showed that the attacks can be detected.

We hope that our methods will be of help in securing vehicles against cyberattacks. Further evaluations should be conducted with much more CAN data. We should also apply our methods to other protocols.

# References


[1] Charlie Miller and Chris Valasek. Remote Exploitation of an Unaltered Passenger Vehicle. In Blackhat 2015, 2015.

[2] The United Nations Economic Commission for Europe (UNECE). World Forum for Harmonization of Vehicle Regulations (WP.29), accessed in Sep. 12, 2020. http://www.unece.org/trans/main/wp29/meeting_docs_grva.html.

[3] Society of Automotive Engineers. Road Vehicles - Cybersecurity Engineering, accessed in Sep. 19, 2020. https://www.sae.org/standards/content/iso/sae21434.d1/.

[4] ISO/SAE DIS 21434. Road vehicles – Cybersecurity engineering, accessed in Sep. 19, 2020. https://www.iso.org/standard/70918.html.

[5] ISO 26262-1:2018. Road vehicles – Functional safety, accessed in Sep. 19, 2020. https://www.iso.org/standard/68383.html.

[6] Armor Defense Inc. Cybersecurity Best Practices: Layered Security, accessed in Sep. 19, 2020. https://www.armor.com/resources/cybersecurity-best-practices-layered-security/.

[7] Priya Dialani. A Layered Approach is Must for Cybersecurity, accessed in Sep. 19, 2020. https://www.analyticsinsight.net/a-layered-approach-is-must-for-cybersecurity/.

[8] U. S. Department of Homeland Security. Recommended Practice: Improving Industrial Control System Cybersecurity with Defense-in-Depth Strategies, 2016. https://us-cert.cisa.gov/sites/default/files/recommended_practices/NCCIC_ICS-CERT_Defense_in_Depth_2016_S508C.pdf.

[9] National Institute of Standards and Technology (NIST). Cybersecurity Framework, accessed in Sep. 12, 2020. https://www.nist.gov/cyberframework.

[10] Takeshi Kishikawa, Hideki Matsushima, Tomoyuki Haga, Manabu Maeda, Yuji Umigami, and Yoshihiro Ujiie. In-Vehicle Network System, Electronic Control Unit, and Irregularity Detection Method. In Publication Number WO/2015/170451, International Applications., 2015.

[11] Satoshi Otsuka, Tasuku Ishigooka, Yukihiko Oishi, and Kazuyoshi Sasazawa. CAN Security: Cost-Effective Intrusion Detection for Real-Time Control Systems. In SAE Technical Paper 2014-01-0340, 2014.

[12] Moti Markovitz and Avishai Wool. Field Classification, Modeling and Anomaly Detection in Unknown CAN bus Networks. In Embedded Security in Cars Conference (escar) Europe 2015, 2015.

[13] Takeshi Kishikawa, Manabu Maeda, Junichi Tsurumi, Tomoyuki Haga, Ryota Takahashi, Takamitsu Sasaki, Jun Anzai, and Hideki Matsushima. A Generic CAN Message Field Extraction Method to Construct Anomaly Detection Systems for In-Vehicle Networks. In 2017 Symposium on Cryptography and Information Security (SCIS 2017), 2017.

[14] Brent Stone. Reverse Engineering 17+ Cars in Less Than 10 Minutes. In Def Con 27, 2019.

[15] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage. Experimental Security Analysis of a Modern Automobile. In 2010 IEEE Symposium on Security and Privacy, 2010.

[16] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In 20th USENIX Security Symposium, 2011.

[17] Chris Valasek and Charlie Miller. Adventures in Automotive Networks and Control Units. In Def con 21, 2013.

[18] Aurelien Francillon, Boris Danev, and Srdjan Capkun. Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars. In Cryptology ePrint Archive, 2010. https://eprint.iacr.org/2010/332.

[19] Troy Hunt. Controlling vehicle features of Nissan LEAFs across the globe via vulnerable APIs, accessed in Sep. 12, 2020. https://www.youtube.com/watch?v=Nt33m7G_42Q.

[20] National Institute of Standards and Technology (NIST). Federal Information Processing Standards Publication, accessed in Sep. 12, 2020. https://csrc.nist.gov/publications/fips.

[21] National Institute of Standards and Technology (NIST). NIST Special Publication 800 Series, accessed in Sep. 12, 2020. https://csrc.nist.gov/publications/sp800.

[22] Larry J.Hughes Jr. Internet Security Teqniques. 1995. New Riders Pub.

[23] Cryptography Research and Evaluation Committees (CRYPTREC). CRYPTREC Ciphers List, accessed in Sep. 12, 2020. https://www.cryptrec.go.jp/en/list.html.

[24] National Institute of Standards and Technology (NIST). Cryptographic Module Validation Program (CMVP), accessed in Sep. 12, 2020. https://csrc.nist.gov/projects/cryptographic-module-validation-program.

[25] Information-technology Promotion Agency, Japan（IPA). Japan Cryptographic Module Validation Program (JCMVP), accessed in Sep. 12, 2020. https://www.ipa.go.jp/security/jcmvp/index.html.

[26] Atsuya Shibata. Mechanism of Secure Boot on ubuntu (In Japanese), accessed in Sep. 12, 2020. https://gihyo.jp/admin/serial/01/ubuntu-recipe/0444.

[27] David Kleidermacher. Securing devices in connected-generation by combining of chips and embedded OS In Japanese, accessed in Sep. 12, 2020. https://ednjapan.com/edn/articles/1211/08/news050_2.html.

[28] Information-technology Promotion Agency, Japan（IPA). Secure Programming Course In Japanese, accessed in Sep. 12, 2020. https://www.ipa.go.jp/security/awareness/vendor/programmingv2/.

[29] Microsoft Corporation. The Basic Knowledge of Windows Update, accessed in Sep. 19, 2020. https://support.microsoft.com/ja-jp/help/884099.

[30] Maria Kelebeev. What is the Difference between Layered Security and Defense in Depth?, accessed in Dec. 14, 2020. https://www.mbccs.com/difference-between-layered-security-and-defense-in-depth/.

[31] National Institute of Standards and Technology (NIST). Framework for Improving Critical Infrastructure Cybersecurity, 2018. https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04162018.pdf.

[32] E-safety vehicle intrusion protected applications. EVITA, accessed in Sep. 19, 2020. https://www.evita-project.org/.

[33] Trusted Computing Group. TCG Protection Profile for PC Client Specific TPM 1.2, accessed in Sep. 19, 2020. https://trustedcomputinggroup.org/resource/tpm-1-2-protection-profile/.

[34] Automotive open system architecture. Standards, accessed in Sep. 13, 2020. https://www.autosar.org/standards/.

[35] Japan Automotive Software Platform and Architecture. Activities and Output, accessed in Sep. 13, 2020. https://www.jaspar.jp/english/organizationalStructure.

[36] Atsushi Ohba. Issues and Research Activities of Automotive Security. In JARI Research Journal, 2018. http://www.jari.or.jp/Portals/0/resource/JRJ_q/JRJ20181101_q.pdf.

[37] Manabu Nakano. Approaches for Connected Vehicles Security. In APCOSEC 2013, 2013. https://www.ipa.go.jp/files/000034549.pdf.

[38] IT Security Center, Information-technology Promotion Agency, Japan(IPA). Approaches for Vehicle Information Security - Information Security for "Networked" Vehicles, 2013. https://www.ipa.go.jp/files/000033402.pdf.

[39] Mitsubishi Electric Corporation. Development of Multi-layered Security Technology for In-vehicle Systems (in Japanese), 2019. https://www.mitsubishielectric.co.jp/news/2019/pdf/0122-b.pdf.

[40] ISO 11898. Road vehicles – Controller area network (CAN) –, 2015. https://www.iso.org/standard/63648.html.

[41] Yasuhiko Abe, Seigo Kotani, and Eiichirou Kubota. Security Technology for OTA Software Updates to Maintain Safety of Motor Vehicles. FUJITSU, Vol.70, No.2 (April, 2019), 2019.

[42] FlexRay consortium. The backup of FlexRay consortium Web page, accessed in Sep. 19 2020. https://web.archive.org/web/20121025131337/http://www.flexray.com/.

[43] IEEE. 802.3bw-2015 - IEEE Standard for Ethernet Amendment 1, accessed in Sep. 19 2020. https://standards.ieee.org/standard/802_3bw-2015.html.

[44] ISO 17987-3:2016. Road vehicles – Local Interconnect Network (LIN) –, 2016. https://www.iso.org/standard/61224.html.

[45] MOST Cooperation. MOST Cooperation Web page, accessed in Sep. 19 2020. https://www.mostcooperation.com/.

[46] Jun Yajima. Cyberattack Detection Technologies for Connected Cars. Monthly In-vehicle Technology, 2019.

[47] Jun Yajima. Recent Studies on Security of Vehicles Innovated by IoT Technologies. In Planned Session (Invited Talk) of IEICE General Conference 2020, 2020.

[48] Peter Waszecki, Philipp Mundhenk, Sebastian Steinhorst, Martin Lukasiewycz, Ramesh Karri, and Samarjit Chakraborty. Automotive Electrical and Electronic Architecture Security via Distributed In-Vehicle Traffic

Monitoring. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2017.

[49] Junichi Tsurumi, Takeshi Kishikawa, Takamitsu Sasaki, Ryota Takahashi, Tomoyuki Haga, and Hideki Matsushima. Proposal of Anomaly Detection Method for In-Vehicle Network based on Relation between Flag type Data. In 2017 Symposium on Cryptography and Information Security (SCIS 2017), 2017.

[50] Yoshihiro Hamada, Keigo Yoshida, Naoki Adachi, Shyogo Kamiguchi, Hiroshi Ueda, Yukihiro Miyashita, Yoshikazu Isoyama, and Yoichi Hata. Intrusion Detection for Acyclic Messages in In-Vehicle Network: A Proposal. In Computer Security Symposium 2018 (CSS 2018), 2018.

[51] Tomohiro Date, Mizuki Teshiba, Takaya Ezaki, and Hiroyuki Inoue. Dynamic Rule Generation using Machine Learning on a Security Gateway for In-vehicle LAN. In 2016 Symposium on Cryptography and Information Security (SCIS 2016), 2016.

[52] Ryota Takahashi, Takamitsu Sasaki, Hideki Matsushima, Tomoyuki Haga, Takeshi Kishikawa, and Junichi Tsurumi. Improving Accuracy of Anomaly Detection for Automotive Security by Sand Sprinkled Isolation Forest. In

2017 Symposium on Cryptography and Information Security (SCIS 2017), 2017.

[53] Ryota Takahashi, Junichi Tsurumi, Takeshi Kishikawa, Takamitsu Sasaki, Tomoyuki Haga, and Hideki Matsushima. Evaluation of Anomaly Detection using Sand Sprinkled Isolation Forest in Real Vehicle Data. In 2018 Symposium on Cryptography and Information Security (SCIS 2018), 2018.

[54] Kazuki Iehira, Kento Kanamori, Hiroyuki Inoue, and Kenji Ishida. Extraction of Correlation between In-vehicle Sensor Information Using Pattern Matching for Automatic Generation of Anomaly Detection Rules. In 2018 Symposium on Cryptography and Information Security (SCIS 2018), 2018.

[55] Jun Yajima, Yasuhiko Abe, and Takayuki Hasebe. Proposal of Anomaly Detection Method for Unstable Periodic Messages on In-Vehicle Network. In 2018 Symposium on Cryptography and Information Security (SCIS 2018), 2018.

[56] Jun Yajima, Yasuhiko Abe, and Takayuki Hasebe. Proposal of Anomaly Detection Method "Cumulative Sum Detection" for In-Vehicle Networks. In Embedded Security in Cars Conference (escar Asia 2018), 2018.

[57] Jun Yajima, Yasuhiko Abe, Takayuki Hasebe, and Takao Okubo. Anomaly Detection Method "Cumulative Sum Detection" for In-Vehicle

Networks. Special issue of Intelligent Transportation Systems and Mobile Communication for Realizing Smart Cities, Journal of Information Processing Vol.28, Information Processing Society of Japan, 2020.

[58] Jun Yajima, Takayuki Hasebe, and Yasuhiko Abe. Attack Detection Device and Attack Detection Method. In Japanese Patent Application P2019-12899A, 2019.

[59] Jun Yajima and Takayuki Hasebe. Attack Detection Device and Attack Detection Method. In Japanese Patent Application P2019-126004A, 2019.

[60] Jun Yajima. Attack Detection Device and Attack Detection Method. In Japanese Patent Application P2020-129785A, 2020.

[61] Fujitsu Laboratories Ltd., Fujitsu Limited. Fujitsu Defends In-Vehicle Networks with New Technology to Detect Cyberattacks, 2018. https://www.fujitsu.com/global/about/resources/news/press-releases/2018/0124-02.html.

[62] Chen Yang, Ryo Kurachi, Gang Zeng, and Hiroaki Takada. Schedulability Comparison for CAN Message with Offset: Priority Queue Versus FIFO Queue. In 19th International Conference on Real-Time and Network Systems, Nantes, France, September 2011.

[63] Robert I. Davis, Alan Burns, Reinder J. Brill, and Johan J. Lukkien. Controller Area Network (CAN) Schedulability Analysis: Refuted, revisited and revised. In Real-Time Systems 2007, 2007.

[64] Robert I. Davis and Nicolas Navet. Controller Area Network (CAN) Schedulability Analysis for Messages with Arbitrary Deadlines in FIFO and Work-conserving Queues. In 9th IEEE International Workshop on Factory Communication Systems, 2012.

[65] Jun Yajima, Ikuya Morikawa, Takayuki Hasebe, and Takao Okubo. Extraction Method of Event Based Periodic Messages for CAN Anomaly Detection. In Computer Security Symposium (CSS 2019), 2019.

[66] Jun Yajima, Ikuya Morikawa, and Takao Okubo. A Study on Feature Extraction for Anomaly Detection on CAN. In 2020 Symposium on Cryptography and Information Security (SCIS 2020), 2020.

[67] Miki E. Verma, Robert A. Bridges, and Samuel C. Hollifield. ACTT: Automotive CAN Tokenization and Translation. In 2018 International Conference on Computational Science and Computational Intelligence (CSCI), 2018.

[68] Micro Marchetti and Dario Stabili. READ: Reverse Engineering of Automotive Data Frames. IEEE Transactions on Information Forensics and Security,

Volume 14, No. 4, April 2019, 2019.

[69] Mert D. Pesé, Troy Stacer, C. Andrés Campos, Eric Newberry, Dongyao Chen, and Kang G. Shin. LibreCAN: Automated CAN Message Translator. In The ACM Conference on Computer and Communications Security (CCS) 2019, 2019.

[70] Jun Yajima, Takayuki Hasebe, and Takao Okubo. Data Relation Analysis Method Based on Data Transition for Attack Detection on Vehicle. In 2019 Symposium on Cryptography and Information Security (SCIS 2019), 2019.

[71] Jun Yajima, Takayuki Hasebe, and Takao Okubo. Data Relation Analysis Focusing on Plural Data Transition for Detecting Attacks on Vehicular Network. In The 22nd International Conference on Network-Based Information Systems (NBiS-2019), 2019.

[72] Jun Yajima and Takayuki Hasebe. Message Processing Devices and Message Processing Method. In Japanese Patent Application P2020-113893A, 2020.

[73] Brent Stone. Automated Payload Reverse Engineering Pipeline for the Controller Area Network (CAN) protocol, accessed May 22, 2020. `https://github.com/brent-stone/CAN_Reverse_Engineering`.

[74] Szia Világ. Prius CAN message Identification Table, accessed in Sep. 21, 2020. http://vassfamily.duckdns.org/ToyotaPrius/CAN/PriusCodes.xls.

[75] Tsutomu Matsumoto, Masato Hata, Masato Tanabe, Katsunari Yoshioka, and Kazuomi Oishi. A Method of Preventing Unauthorized Data Transmission in Controller Area Network. In IEEE 75th Vehicular Technology Conference (VTC Spring), 2012.

[76] Taehoon Park. Hierarchical Division of Labor Structure and Inter-organizational Relationship of the Japanese Automobile Industry (in Japanese). Japan Society of Business Administration, 2003.

[77] Jun Yajima, Takayuki Hasebe, Naoya Torii, and Tsutomu Matsumoto. CAN Security System that enables Vehicle Stopping Safely after Detecting Attack. In 2016 Symposium on Cryptography and Information Security (SCIS 2016), 2016.

[78] Masato Tanabe, Yoshihiko Kitamura, Jun Anzai, Takeshi Kishikawa, Yoshihiro Ujiie, Tomoyuki Haga, and Hideki Matsushima. A Secure Switching Method between Monitoring Mode and Verifying Mode for In-Vehicle Network. In 2015 Symposium on Cryptography and Information Security (SCIS 2015), 2015.

# Acknowledgements

This doctoral dissertation consists of a summary of my research-work at the Institute of Information Security (IISEC), Kanagawa, Japan. I really appreciate all person helped to the research-work of this dissertation and the activities of the doctoral program.

First of all, I would like to thank Professor Takao Okubo, my supervisor, for accepting me as a PhD student and for giving me various valuable advice on my research activities.

I also express my appreciation for members of dissertation committee of my phD defense: Professor Toshihiro Matsui, Associate Professor Midori Inaba, and Associate Professor Masaki Hashimoto who have given me many useful advice and suggestion of improving idea of the doctoral dissertation.

I wish to thank my collaborated researchers. Mr. Takayuki Hasebe, my former boss and collaborator in Fujitsu Laboratories Ltd., has supported me on whole my doctoral research. I have discussed him many times about vehicle security and

have been inspired through the discussions.

Mr. Ikuya Morikawa, my current boss in Fujitsu Laboratories Ltd., has supported me as a current boss in my company. He has understood my activities in the doctoral program. And he has giving me many advices in discussions, especially proposal 2.

Mr. Hisashi Kojima, my previous boss in Fujitsu Laboratories Ltd., allowed me to enroll the graduate university.

Mr. Yasuhiko Abe, Expert in Fujitsu Limited, has supported me in many discussions, especially proposal 1. He gave me advices on the application of development technology.

Mr. Ryuichi Ohori, Researcher in Fujitsu Laboratories Ltd., has supported me on some mathematical verification in this doctoral dissertation.

I wish to thank university instructors in IISEC. President Atsuhiro Goto, Professor Toshihiro Matsui, and Professor Akira Otsuka taught me various specialized knowledge in compulsory class in the doctoral program.

I express my appreciation to coworkers in Security Laboratory in Fujitsu Laboratories Ltd. They has understood my doctoral activities and supported me in various research-life.

I also express my appreciation to members of Okubo Laboratory in IISEC. They have given me many various comments to my presentations in seminar. I could have enjoyed studying in IISEC by pleasant conversations with them.

I have special appreciation to Dr. Shigeo Tsujii who was a Professor at my master program in Chuo University and was also the first president of IISEC. He taught me foundation knowledge of information security and led me to information security field as a researcher.

Last but not least, I also would like to thank my friends, my sister and her family for spending happy life with me. I am very much to thankful my parents, father Hitoshi and mother Mitsue, for their understanding, encouragement, and perpetual support.

Jun Yajima

March 2021.

# List of Publications Related to the Dissertation

## Journal papers

1. Jun Yajima, Yasuhiko Abe, Takayuki Hasebe, and Takao Okubo, "Anomaly Detection Method "Cumulative Sum Detection" for In-Vehicle Networks," Special Issue of Intelligent Transportation Systems and Mobile Communication for Realizing Smart Cities, Journal of Information Processing Volume 28, Information Processing Society of Japan (IPSJ), pp.65-74, 2020.

## Conference Papers

1. Jun Yajima, Yasuhiko Abe, and Takayuki Hasebe, "Proposal of Anomaly Detection Method "Cumulative Sum Detection" for In-Vehicle Networks," The Embedded Security in Cars Conference (escar) Asia 2018, 2018.

2. Jun Yajima, Takayuki Hasebe, and Takao Okubo, “Data Relation Analysis Focusing on Plural Data Transition for Detecting Attacks on Vehicular Network,” The 22nd International Conference on Network-Based Information Systems (NBiS-2019), Advances in Networked-based Information Systems, pp.270-280, 2020.

# List of All Publications

# (Main Works)

## Journal papers

1. Jun Yajima, Yasuhiko Abe, Takayuki Hasebe, and Takao Okubo, "Anomaly Detection Method "Cumulative Sum Detection" for In-Vehicle Networks," Special Issue of Intelligent Transportation Systems and Mobile Communication for Realizing Smart Cities, Journal of Information Processing Volume 28, Information Processing Society of Japan (IPSJ), pp.65-74, 2020.

2. Jun Yajima, Terutoshi Iwasaki, Yusuke Naito, Yu Sasaki, Takeshi Shimoyama, Thomas Peyrin, Noboru Kunihiro, and Kazuo Ohta, "A Strict Evaluation on the Number of Conditions for SHA-1 Collision Search," IEICE Transactions 92-A(1): 87-95 2009, 2009.

## Conference Papers

1. Jun Yajima, Takayuki Hasebe, and Takao Okubo, “Data Relation Analysis Focusing on Plural Data Transition for Detecting Attacks on Vehicular Network,” The 22nd International Conference on Network-Based Information Systems (NBiS-2019), Advances in Networked-based Information Systems, pp.270-280, 2020.

2. Jun Yajima, Yasuhiko Abe, and Takayuki Hasebe, “Proposal of Anomaly Detection Method “Cumulative Sum Detection” for In-Vehicle Networks,” The Embedded Security in Cars Conference (escar) Asia 2018, 2018.

3. Jun Yajima, and Takeshi Shimoyama, “Matrix Representation of Conditions for the Collision Attack of SHA-1 and Its Application to the Message Modification,”International Workshop on Security (IWSEC) 2010, 2010.

4. Jun Yajima, Terutoshi Iwasaki, Yusuke Naito, Yu Sasaki, Takeshi Shimoyama, Noboru Kunihiro, and Kazuo Ohta, “A Strict Evaluation Method on the Number of Conditions for the SHA-1 Collision Search,” ACM Asia Conference on Computer & Communications Security (AsiaCCS) 2008, 2008.

5. Jun Yajima, Yu Sasaki, Yusuke Naito, Terutoshi Iwasaki, Takeshi Shimoyama, Noboru Kunihiro, and Kazuo Ohta, “A New Strategy for Finding a Differential Path of SHA-1,” Australasian Conference on Information Security and

Privacy (ACISP) 2007, 2007.

## Preprints

1. Jun Yajima, and Takeshi Shimoyama, “Wang’s sufficient conditions of MD5 are not sufficient,” Cryptology ePrint archive, 2005.

## Articles

1. 矢嶋純, “コネクテッドカーに対するサイバー攻撃の検知技術,” 月刊車載テクノロジー 2019 年 6 月号, 技術情報協会, pp.8-11, 2019.（執筆依頼）

## Presentations

1. 矢嶋純, “IoT 化が進みつつある自動車におけるセキュリティの現状 (2),” 2021 年電子情報通信学会総合大会企画セッション「IoT システムにおけるハードウェアセキュリティの最新動向」, 2021.

2. 矢嶋純, 清水俊也, 森川郁也, 大久保隆夫, “機械学習に潜む AI セキュリティ脆弱性の分析手法に関する一考察,” 2021 年暗号と情報セキュリティシンポジウム (SCIS), 2021.

3. 矢嶋純, “IoT 化が進みつつある自動車におけるセキュリティの現状,” 2020 年電子情報通信学会総合大会企画セッション「IoT システムにおけるハードウェアセキュリティ最前線」, 2020.

4. 矢嶋純, 森川郁也, 長谷部高行, 大久保隆夫, “CAN のイベント送信付き周期メッセージの検出と攻撃検知への応用,” コンピュータセキュリティシンポジウム（CSS）2019, 2019.

5. 矢嶋純, 森川郁也, 大久保隆夫, “CAN の攻撃検知における特徴量抽出に関する一考察,” 2020 年暗号と情報セキュリティシンポジウム (SCIS), 2020.

6. 矢嶋純, 長谷部高行, 大久保隆夫, “値の遷移に着目した車載向け攻撃検知のためのデータ関連性分析手法,” 2019 年暗号と情報セキュリティシンポジウム (SCIS 2019), 2019.

7. 矢嶋純, 阿部保彦, 長谷部高行, “車載ネットワークの周期乱れ発生時にも高精度に攻撃を検知可能な累積和検知方式の提案” 2018 年暗号と情報セキュリティシンポジウム (SCIS 2018), 2018.

8. 矢嶋純, 長谷部高行, “CAN の周期送信メッセージに対する攻撃検知手法の詳細評価とその評価手法,” 2017 年暗号と情報セキュリティシンポジウム (SCIS 2017), 2017.

9. 矢嶋純, 長谷部高行, 鳥居直哉, 松本勉, “「攻撃メッセージの無効化機能を備えたホワイトリスト CAN ハブ」の実装評価，及び，エラーフレームによる無効化機能を用いたホワイトリスト CAN ハブの提案,” 2016 年暗号と情報セキュリティシンポジウム (SCIS 2016), 2016.

10. 矢嶋純, 長谷部高行, 鳥居直哉, 松本勉, “攻撃検知後の自動車の安全な停車を可能にするCANセキュリティシステム,” 2016年暗号と情報セキュリティシンポジウム (SCIS 2016), 2016.

11. 矢嶋純, 長谷部高行, “非周期送信メッセージによる攻撃を検知可能にするセキュリティCANアダプタ,” 2016年暗号と情報セキュリティシンポジウム (SCIS 2016), 2016.

12. 矢嶋純, 武仲正彦, 長谷部高行, “攻撃メッセージの無効化機能を備えたホワイトリストCANハブ,” 2015年暗号と情報セキュリティシンポジウム (SCIS 2015), 2015.

13. 矢嶋純, 安田雅哉, 下山武司, 小暮淳, “Gentry準同型暗号に対するBKZ攻撃実験,” 2012年暗号と情報セキュリティシンポジウム (SCIS 2012), 2012.

14. 矢嶋純, 安田雅哉, 下山武司, 小暮淳, “Gentry準同型暗号に対するLLL攻撃実験(II) ,” 電子情報通信学会情報セキュリティ研究会 (ISEC) 2011, 2011.

15. 矢嶋純, 安田雅哉, 下山武司, 小暮淳, “Gentry準同型暗号に対するLLL攻撃実験について,” コンピュータセキュリティシンポジウム2011 (CSS 2011), 2011.

16. 矢嶋純, 下山武司, “SHA-1のコリジョン探索におけるコンディションの

行列表示とメッセージモディフィケーションへの応用," 2010 年暗号と情報セキュリティシンポジウム (SCIS 2010), 2010.

17. 矢嶋純, 下山武司, "SHA-1 型ハッシュ関数の安全性評価ツールの設計,"2009 年暗号と情報セキュリティシンポジウム (SCIS 2009), 2010.

18. 矢嶋純, 下山武司, "SHA-1 のコリジョン探索における Message Modification 適用可否判定法," 2008 年暗号と情報セキュリティシンポジウム (SCIS 2008), 2008.

19. 矢嶋純, 佐々木悠, 岩崎輝星, 内藤祐介, 下山武司, 國廣昇, 太田和夫, "SHA-1 差分パス自動生成ツール," 2007 年暗号と情報セキュリティシンポジウム (SCIS 2007), 2007. (SCIS 論文賞受賞)

20. 矢嶋純, "ハッシュ関数のコリジョン探索攻撃の現状," 金融リスク管理のための新ＩＴモデルの研究と開発（科研費シンポジウム）, 2006.（招待講演）

21. Jun Yajima, Yu Sasaki, Terutoshi Iwasaki, Yusuke Naito, Takeshi Shimoyama, Noboru Kunihiro, and Kazuo Ohta, "Constructing differential paths for SHA-1 collision attack," Rump Session of the 26th Annual International Cryptology Conference (CRYPTO) 2006, 2006.

22. 矢嶋純, 下山武司, 佐々木悠, 内藤祐介, 國廣昇, 太田和夫, "MD5 のコリジョン探索における差分パスの構築法について," 2006 年暗号と情報セ

キュリティシンポジウム (SCIS 2006), 2006.

23. 矢嶋純, 下山武司, “MD5 のコリジョン探索および Sufficient Conditions について” , 電子情報通信学会情報セキュリティ研究会 (ISEC) 2005, 2005.

24. 矢嶋純, 武仲正彦, 下山武司, “共通鍵暗号モジュールの試験に関する一考察,” 電子情報通信学会情報セキュリティ研究会 (ISEC) 2004, 2004.

25. 矢嶋純, 伊藤孝一, 武仲正彦, 鳥居直哉, “Window method の改良による公開鍵暗号への DPA 対策,” 2002 年暗号と情報セキュリティシンポジウム (SCIS 2002), 2002.

26. 矢嶋純, 武仲正彦, 小柴健史, 鳥居直哉, “共通鍵ブロック暗号 SC2000 の乱数性,” 電子情報通信学会情報セキュリティ研究会 (ISEC) 2001, 2001.

27. 矢嶋純, 武仲正彦, 鳥居直哉, “Serpent の S-box における効率的な計算法について,” 2000 年電子情報通信学会総合大会, 2000.

28. 矢嶋純, 武仲正彦, 鳥居直哉, “Serpent における S-box の効率的な論理表現に関する考察,” 2000 年暗号と情報セキュリティシンポジウム (SCIS 2000), 2000.

29. 矢嶋純, 下山武司, 辻井重男, “SERPENT-FSE 版と AES 版の強度比較,” 1999 年暗号と情報セキュリティシンポジウム (SCIS 1999), 1999.

30. 矢嶋純, 下山武司, 辻井重男, “共通鍵ブロック暗号 SERPENT(AES 候補)

のラウンド関数の高階差分について," 電子情報通信学会情報セキュリティ研究会 (ISEC) 1998, 1998.

# Paper Reuse Permission

I received the paper reuse permission from the following organizations and companies.

- Information Processing Society of Japan (IPSJ)
- Copyright Clearance Center
- Nikkei Automotive Seminar Office

# Author Biography

Jun Yajima was born in 1974. He received his B.E. and M.E. in Information and System Engineering from Chuo University in 1997 and 1999, respectively. He joined Fujitsu Laboratories Ltd in 1999. His current research interests include vehicle security, machine learning security, cryptography, and implementation technology of cryptography. Since April 2019, he has been working towards PhD under the supervision of Professor Takao Okubo, at the Graduate School of Information Security, Institute of Information Security, Kanagawa, Japan. He was awarded the SCIS2007 paper prize and the SCIS2012 innovation paper prize. He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE) of Japan. He is also a member of the Information Processing Society of Japan (IPSJ). Since 2017, He is a member of IEICE Technical Committee on Hardware Security (HWS). He is a member of the working group on JCMVP Cryptographic Algorithm Implementation Testing Requirements, Information-technology Promotion Agency (IPA), Japan.