Master's Thesis
석사 학위논문

A Practical and Lightweight Source Authentication Protocol Using One-Way Hash Chain in CAN

Ki-Dong Kang(강 기 동 姜 基 東)

Department of Information and Communication Engineering

정보통신융합전공

DGIST

2017

Master's Thesis
석사 학위논문

A Practical and Lightweight Source Authentication Protocol Using One-Way Hash Chain in CAN

Ki-Dong Kang(강 기 동 姜 基 東)

Department of Information and Communication Engineering

정보통신융합전공

DGIST

2017

A Practical and Lightweight Source Authentication Protocol Using One-Way Hash Chain in CAN

Advisor: Professor Sang Hyuk Son

Co-Advisor: Ph.D. Seonghun Lee

Co-Advisor: Ph.D. Youngmi Baek

by

Ki-Dong Kang

Department of Information and Communication Engineering

DGIST

A thesis submitted to the faculty of DGIST in partial fulfillment of the requirements for the degree of Master of Science in the Department of Information and Communication Engineering. The study was conducted in accordance with Code of Research Ethics[1]

25. 11. 2016

Approved by

Professor (Advisor)	Sang Hyuk Son	(Signature)
Ph.D. (Co-Advisor)	Seonghun Lee	(Signature)
Ph.D. (Co-Advisor)	Seonghun Lee	(Signature)

[1] Declaration of Ethical Conduct in Research: I, as a graduate student of DGIST, hereby declare that I have not committed any acts that may damage the credibility of my research. These include, but are not limited to: falsification, thesis written by someone else, distortion of research findings or plagiarism. I affirm that my thesis contains honest conclusions based on my own careful research under the guidance of my thesis advisor.

A Practical and Lightweight Source Authentication Protocol Using One-Way Hash Chain in CAN

Ki-Dong Kang

Accepted in partial fulfillment of the requirements
for the degree of Master of Science.

25. 11. 2016

MS/IC
201522001

강 기 동. Ki-Dong Kang. A Practical and Lightweight Source Authentication Protocol Using One-Way Hash Chain in CAN. Department of Information and Communication Engineering. 2016. 37p. Advisor Prof. Sang Hyuk Son, Co-Advisor Ph.D. Seonghun Lee, Co-Advisor Ph.D. Youngmi Baek.

ABSTRACT

While vehicle to everything (V2X) communication enables safety-critical automotive control systems to better support various connected services to improve safety and convenience of drivers, they also allow automotive attack surfaces to increase dynamically in modern vehicles. Many researchers as well as hackers have already demonstrated that they can take remote control of the targeted car by exploiting the vulnerabilities of in-vehicle networks such as Controller Area Networks (CANs). For assuring CAN security, we focus on how to authenticate electronic control units (ECUs) in real-time by addressing the security challenges of in-vehicle networks. In this thesis, we propose a novel and lightweight authentication protocol with an attack-resilient tree algorithm, which is based on one-way hash chain. The protocol can be easily deployed in CAN by performing a firmware update of ECU. We have shown analytically that the protocol achieves a high level of security. In addition, the performance of the proposed protocol is validated on CANoe simulator for virtual ECUs and Freescale S12XF used in real vehicles. The results show that our protocol is more efficient than other authentication protocol in terms of authentication time, response time, and service delay.

Keywords: Controller Area Network, In-Vehicle Network Security, Authentication, Cyber-Physical Systems (CPS)

Contents

List of Figures

List of Tables

I. Introduction

Recently, automotive cyber-physical systems (CPS) are getting attention as one of novel paradigms to deal with connected car. Although they provide great potential for vehicle safety, convenience, and efficiency, they cause the automotive system to be more complicated and might expose the vulnerability outside. In fact, researchers have demonstrated that they can take remote control of the targeted car by conducting experiments with various methods [1, 2]. Behind these attacks, there are security vulnerabilities of Controller Area Network (CAN) [3] which is a de facto standard for in-vehicle networks to exchange signals among Electronic Control Units (ECUs). Critical vulnerabilities of CAN include no identification mechanism (no address of the sender or receiver) and a payload size being too small for authentication.

To address these vulnerabilities, prior research mainly investigates authentication methods to prevent cyber-attacks in CAN. They are typically carried out in centralized or distributed fashion. Since centralized authentication methods typically follow a master-slave model, a master node not only has to bear associated computational overhead but also consumes additional bandwidth to distribute keys and authenticate slaves [5, 26]. In this regard, it does not fit well in CAN environments. Distributed authentication is an approach in which each node has the capability to verify a message if it is sent by a trusted sender [4, 6]. While it performs a complicated key management scheme among nodes, it does not need additional messages for authentication.

Authentication for verifying the true identity of ECUs is performed by using an authentication value (an authenticator) which is appended to a message. To generate an authenticator, Message Authentication Code (MAC) and hash function [4, 5, 6, 11, 13, 14, 27] are typically used. Although MAC guarantees the desired authentication level, it usually requires high computing power in low-performance ECUs connected by CAN. A one-way hash chain is one of the classical hash function methods for authentication with insecure communications. It is hardly exploited in CAN since it has a hash collision problem (i.e., a vulnerability for hash collision attacks). To address the hash collision problem, we propose an attack-resilient algorithm to be included in our authentication protocol using one-way hash chain for lightweight authentication based on distributed authentication.

Since the authentication protocol using one-way hash function needs a shared secret key to generate a series of authenticators, the range of key sharing is important. There are three methods for the range of secret key sharing: a session key, a pair-wise key, and a group key. Each method has its advantages and disadvantages. In the session key method, one or more nodes use one pre-shared key for communications and keep it for one session. This method is easy to manage the key since they need to distribute only one session key [6, 11]. Communications among nodes with the same session key, however, is delayed during a specific time period in which it is updated because the session key must be updated periodically in order to reduce the risk of attacks. In addition, as the nodes with the same session key increase, the delay caused by key update increases. Since CAN does not accept authentication delay of messages related to safety-critical controls, it is not appropriate for CAN. In the pair-wise key method, a pair of nodes (a sender and a receiver) uses a specific pre-shared key [4, 13, 14]. The sender needs to manage lots of associated keys as the number of pairs in the entire network increases. It should use a different key for each receiver because different keys are used for a pair of nodes. Sharing the pair-wise key is proposed for a certain group where receivers use split keys assigned by a sender for authentication [5]. However, there exists an overhead for the sender to manage the group and split keys for receivers. Further, it requires additional messages for authentication of each message. In the group key method, a sender shares one secret key with one or more receivers and is usually effective for the applications requiring multicast. Since many messages from senders are broadcast to the entire network or to predefined receivers in CAN, a group key can be easily used in point (a sender) to multipoint (receivers) communications. Therefore, we utilize the concept of group key in order to minimize delays caused by the key update and decrease the number of secret keys.

In this thesis, we propose a novel authentication protocol which utilizes the one-way hash chain using a sender-based group key for CAN. In the whole network, a group can consist of the sender and receivers with the specific shared keys during a given time period. Consequently, the receivers in the group can verify messages with the authenticator received from the alleged sender. We validate that the proposed protocol achieves better performance than an existing protocol in terms of authentication time, response time, and service delay. It is suitable to the CAN environment due to little impact on authentication time. Further, it is a practical solution since it is easy to be deployed by updating the firmware of ECUs.

The remainder of this thesis is organized as follows. In Section II, we describe the background and challenges of authentication in CAN environments. Section III provides the detail description of our proposed protocol and algorithm. In Section IV, we analyze the security level of the proposed protocol. In Section V, we show experimental results. Finally, the thesis is concluded with future work in Section VI.

II. Background and Challenges

2.1 Challenges for Enhanced Security in In-Vehicle Networks

The CAN is a broadcast communication protocol designed by Robert Bosch GmbH in 1986 for in-vehicle networks and currently used as a vehicle bus standard [3]. Since CAN consists of CAN High and CAN Low and uses the difference between them for communication, it provides a highly stable channel. It is estimated that one undetected error may happen every 1000 years in the CAN environment [9].

There are two types of CAN, which are CAN 2.0A and CAN 2.0B. The difference between them is only the length of message ID, 11 and 29 bits, respectively. We basically consider CAN 2.0B for our authentication protocol in order to utilize extended ID field (18 bits).

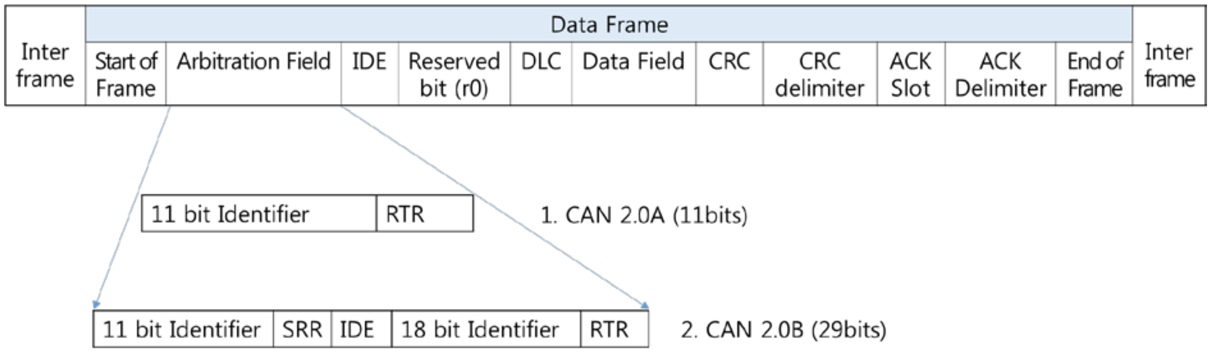


Figure 1. Data frame structure in CAN protocol

As shown in Figure 1, since CAN does not provide any information about a sender, the receivers cannot identify each message coming from the trusted sender. It is especially vulnerable to spoofing attacks including masquerade attacks and replay attacks. Limitations of CAN for security services include

- The length of the data field is maximum 64 bits.
- Limited bandwidth utilization (Maximum baud rate is 1 Mbps in high speed CAN).
- Broadcast nature.
- Low performance of current ECUs for strong cryptographic operations.
- No fields related to security in a data frame.

2.2 Existing Methods for Security

Although it is not easy to provide security for the CAN environment because of the challenges discussed above, a number of security protocols are proposed in order to offer authentication for CAN.

Bogdan *et al*. [5] have proposed an authentication method based on a split key for CAN. Because of its complicated authentication process, it causes bandwidth utilization overhead and the number of ECUs using the proposed method is limited. Lin *et al*. [4] have presented the security protocol using message ID tables, pair-wise secret keys, and message-based counters. But it also demands too much bandwidth utilization and is restricted in the broadcast environment of CAN. The study of Samuel *et al*. [6] shows the performance of a security protocol providing both encryption and authentication with a novel attack scenario. However, it should not only modify the CAN protocol but also have high-end chips to perform well. In Europe, there is E-safety

Vehicle Intrusion protected Applications (EVITA) project [10] handling security issues of on-board networks. The noticeable result of this project is Hardware Security Module (HSM). However, it not only requires additional hardware but also does not provide a specific solution which can be applicable [7]. Apart from the design of the security protocol working on CAN, the effort to integrate a security function into the CAN standard protocol has been made [13]. It allows MAC to be divided into four fragmented parts in order to use the CRC field of a data frame. This method is not suitable for CAN because authentication is delayed due to a lot of small fragments of MAC putting into the limited CRC field. It also needs to modify CAN standard. Lastly, an optimization method for including MAC into the payload of a data frame is proposed in [14]. However, they still suffer from a short length of MAC and payload.

Many authentication protocols have been generally designed using One Time Password (OTP) [15] since it can be very useful in the constraint environment such as embedded systems of automobiles or wireless sensor networks. There are four common methods in OTP (i.e., the event synchronization method, the challenge-response method, the time synchronization method, and the S/KEY method).

The event synchronization method can be used in CAN [4, 6] because it makes a protocol simple and easy to implement. The event synchronization method counts the number of event occurrences using a counter. However, it achieves a low authentication level since the event can be easily predicted by adversaries who can eavesdrop the network. If authentication protocols leverage the event synchronization method, they should guarantee the desired authentication level.

Although the challenge-response method is very popular with RFID fields [17, 18], it is not suitable for the CAN environment since it should perform at least 2 or 3-way handshake for every authentication process. Such high overhead caused by the handshake limits its applicability to CAN.

For wireless sensor network, there is a TESLA broadcast authentication protocol using the time synchronization method [12]. The TESLA needs to transmit a seed from a sender to one or more receivers in order to share seeds, which is conducted based on the S/KEY method. It is referred to as a hybrid protocol since it uses both the time synchronization method and the S/KEY method. Its authentication processing performance is constantly maintained regardless of the number of members. Specially, Bogdan *et al*. try to apply it for the CAN environment [8]. However, the time synchronization method is not appropriate for the CAN environment because of two reasons: (1) it needs centralized global time synchronization devices (e.g., a central gateway has a role of global time master) (2) the authentication protocol using the time synchronization method such as TESLA leads to a long authentication delay.

The S/KEY method is to construct the one-way hash chain and then use it in the opposite direction as an authenticator [16, 19]. If it uses the one-way hash function for a series of authentication, it can provide efficient authentication in terms of the computational cost and a high security level.

2.3 Problems of the One-Way Hash Chain

A key characteristic of the one-way hash chain is that it uses hash values in the opposite direction to the generation of hash values. Due to this feature, after the hash chain is once used in authentication during run time, it is never regenerated using the same seed. In addition, all hash functions basically suffer from collisions since the hash functions can make the same output from different inputs. It must be solved to utilize the one-way hash function for authentication. In the general IT environment, the probability of collision is negligible as they can take a sufficient payload for authentication. However, it is challenging to handle it in the CAN environment having only 64-bit length as a payload. In order to adapt the one-way hash chain to CAN, a novel authentication protocol should address three crucial issues as the following:

1) How to minimize a delay for authentication: If the hash chain is exhausted, it generally takes significant time to generate the hash chain again.

2) How to securely share a seed for re-generation of the hash chain.

3) How to support robustness against hash collisions: Hash collision attacks by adversaries can occur in any receiver using the one-way hash chain because of a short length of payload in the data frame.

III. Source Authentication Protocol

In this section, we present a lightweight authentication protocol, called SAP (Source Authentication Protocol). SAP allows ECUs to use one-way hash chain to defend CAN against spoofing attacks. The characteristics of SAP are: (1) a sender-based group key method, (2) lightweight authentication and key update procedure for CAN, and (3) easy to deploy by ECUs' firmware update.

3.1 Assumption and Attack Model

The group key distributed by a certain sender is only shared with the members of a group which is defined as a set of one sender and associated receivers. In CAN, one or more receivers are generally interested in messages transmitted by the sender. We assume that during the system initiation phase, members of a group are given both the long-term symmetric key for seed sharing and an ID of the sender as mentioned above. An ECU as a sender needs to generate and maintain a table of hash values (i.e., authenticator). Each hash value is inserted in a data frame for transmission after the sender takes it out of the hash table. We consider CAN 2.0B for our authentication protocol in order to utilize extended ID field (18 bits) for inserting authenticator. We do not consider self-collision during constructing hash chain step since this problem can be solved easily through some techniques such as shifting the hash value. Lastly, a new random nonce as a seed value in this protocol is distributed to members of the group by the sender before consuming all hash values from the hash table.

The attack model we consider in this thesis is that adversaries perform replay attacks or masquerade attacks by reading and inserting CAN data frames. They can attack ECUs on CAN through Firmware update Over The Air (FOTA) [23, 24] or connection of OBD2 port with malicious smartphone applications [6]. Table 1 presents our attack model based on the Computer Emergency Response Team (CERT) taxonomy [20]. The detailed scenarios related to our work are referred in [6, 20, 24] and we do not consider Denial of Service (DoS) attack.

Table 1. CERT classification of the proposed attack model

Attacker	Tool	Vulnerability	Action	Target	Unauthorized result	Object
Hackers	Firmware update Over The Air, Malicious smartphone app	Design of CAN	Read, Spoof	Vehicle with ECUs	ECU forced control	Challenge, Thrill

3.2 Proposed Authentication Protocol

SAP consists of three phases: (1) initialization, (2) transmission, and (3) seed value sharing. The list of notations regarding SAP is given in Table 2. The overall concept of SAP is shown in Figure 2.

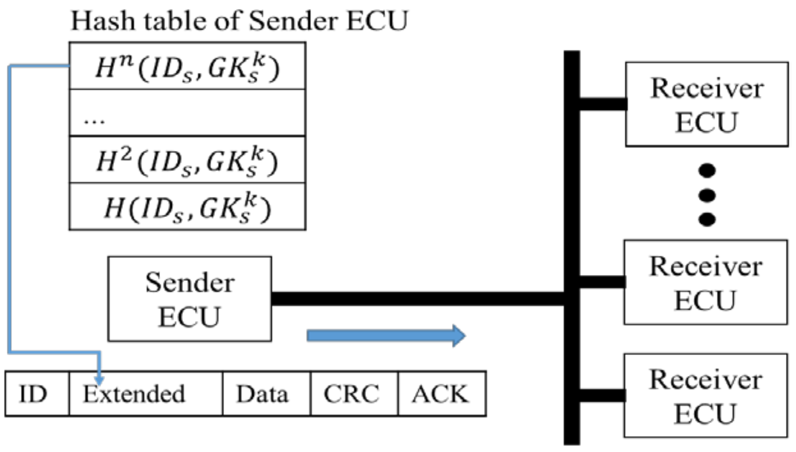


Figure 2. Overall concept of SAP

Table 2. Notation used in this thesis

Notation	Description
ID_s	ID of a sender ECU
K_s	Long-term symmetric key of a sender ECU
$Seed_s^k$	Seed value of k_{th} hash table of a sender ECU
EK_s^k	Encryption key of k_{th} hash table of a sender ECU
$E_{EK}()$	Encryption function using encryption key
C	Ciphertext
GK_s^k	Group key for k_{th} hash table of a sender ECU
$KDF()$	Key derivation function using one-way function
$H()$	One-way hash function $\text{H}: \{0,1\}^* \rightarrow \{0,1\}^{18}$
$H^i()$	i times nested hash function $(0 < \text{i} \leq \text{n})$ (e.g., $H^i(\text{ID}, \text{GK}) = H(H^{i-1}(ID, GK), ID, GK)$, $H^0 = \emptyset$ and $H^1(ID, GK) = H(ID, GK)$)
n	Maximum number of hash value in hash table

1) *Initialization phase:* In this phase, we utilize well-known techniques, which are the long-term symmetric key and Authenticated Key Exchange Protocol 2 (AKEP 2) for a secure exchange of seeds. For these techniques, the long-term symmetric key (K_s) and the ID (ID_s) of the sender should be stored during a manufacturing step of ECU. Since these techniques are beyond of the scope of this thesis, we omit the detail explanation of them. As a result of this phase, each sender ECU securely shares the first seed value ($Seed_s^1$) with receiver ECUs in the same group for generating the group key (GK_s^1), and then construct the one-way hash table using it. At the same time, the receiver ECUs generate the last hash values (*n+1*th value) from the one-way hash chain using the first seed value, where the number of hash values in the hash table of the sender is *n*. This is because the *n+1*th value are only used to perform the *n*th hashed value of the sender ECU through one-way hash chain.

Note that, instead of storing all of the hash values of the sender, a receiver ECU stores at least one hash value for the sender. As the number of hash values in the receiver ECU is directly associated with the robustness against hash collision attacks, it is important to use the appropriate number of hash values. It is discussed in the next subsection.

If the sender ECU needs a new group key (GK_s^2), the next seed value ($Seed_s^2$) of the sender is securely shared within the group by using long-term symmetric key and AKEP 2 or through seed value sharing phase as described below. Figure 3 shows the result of this phase.

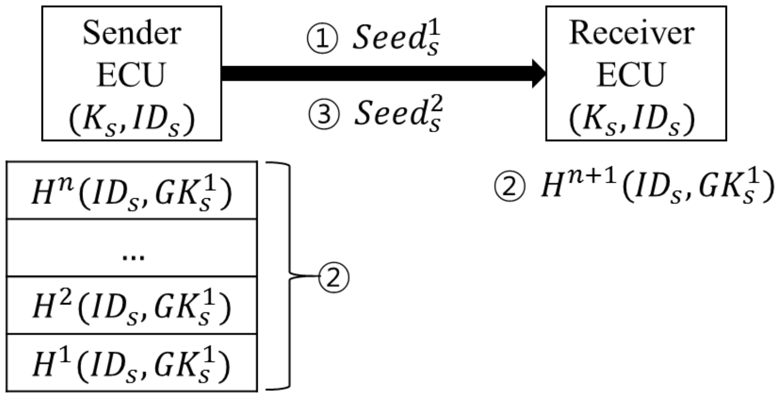


Figure 3. The stored values after finishing initialization phase

2) Transmission phase: In SAP, a sender ECU constructs a hash chain and uses a hash value from the hash chain as shown in Figure 2. Note that we use the one-way hash chain as shown in Figure 4. For authentication, the hash chain is formed by applying the one-way hash function *n* times, where *n* is the length of the one-way hash chain. The sender uses the hash value in the opposite order for source authentication. Because of the one-way property, an adversary cannot predict what the next hash value is.

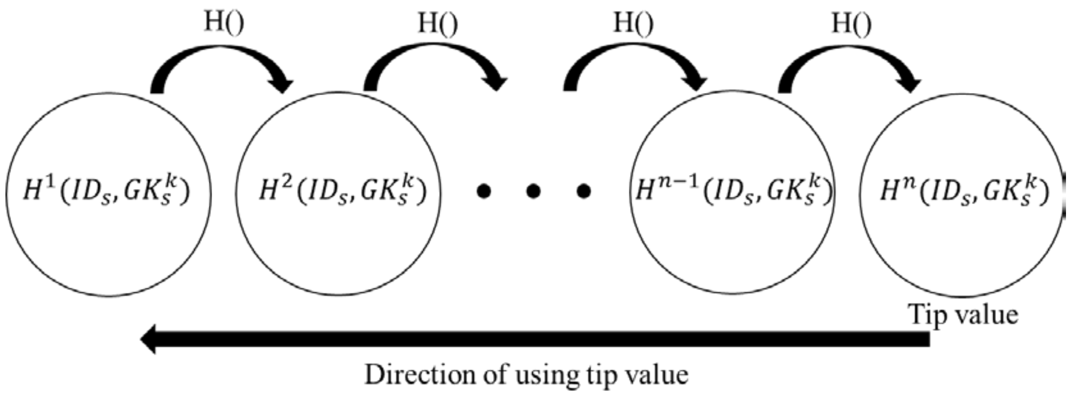


Figure 4. One-way hash chain for SAP

If a sender ECU consumes all hash values from the hash table after it constructs the first hash chain by using the group key in the initialization phase, the sender ECU needs to re-construct hash chain. This results in large delay for a new one-way hash chain as described in Section 2.3. In order to prevent this problem, each sender ECU generates one new hash value for the hash chain using the next value (GK_s^{i+1} , the current hash table uses GK_s^i) at every transmission. Each receiver ECU also generates one new hash value for the hash chain in every reception. Since the receiver ECU authenticates the source of each CAN message using one-way hash chain, it needs to store the *n+1*th hash value to authenticate the *n*th hash value from the sender ECU. The detail of this phase is illustrated below.

Transmission phase

```
Sender ECU                                                    Receiver ECU
Send a CAN message including tip value
(tip value = $H^i(ID_s, GK_s^k)$)
Generate a hash value for next hash table
($hash\ value\ for\ next\ hash\ table = H^{n-i}(ID_s, GK_s^{k+1})$)
                                      Receive a CAN message including tip value
                                      Hash the received tip value
                                      Compare stored hash value with the hashed tip value
                                      ($H(H^i(ID_s, GK_s^k)) = H^{i+1}(ID_s, GK_s^k)$)
                                      if false then
                                          Authentication fail
                                      else then
                                          Store $H^i(ID_s, GK_s^k)$ as a next hash value
                                          Generate a hash value for next hash table
```

3) Seed value sharing phase: Before exhausting all hash values in the hash table, each sender ECU must share the next seed value to receiver ECUs belonging to the same group. This phase is related to the second problem in Section 2.3. After receiving the seed value from a sender ECU, each receiver ECU derives a new key using Key Derivation Function (KDF) for next hash table. The detail of this phase is as follows.

Seed value sharing phase

```
Sender ECU                                                        Receiver ECU
Generate a seed value for the $k+1_{th}$ group hash table
($seed_s^{k+1}$)
Encrypt a previous seed value using an encryption key
($E_{EK_s^k}(seed_s^k)$)
Generate C using the encrypted seed and the new seed
($C = E_{EK_s^k}(seed_s^k) \oplus seed_s^{k+1}$)
Send a CAN message including C and a tip value
Generate a hash value for the next hash table
Derive a group key for the next hash table and an encryption key
($KDF(K_s, ID_s, seed_s^{k+1}) = GK_s^{k+1} || EK_s^{k+1}$)
                                        Receive a CAN message including C and a tip value
                                        Hash the received tip value
                                        Compare a stored hash value with the hashed tip value
                                        if false then
                                            Authentication fail
                                        else then
                                            Obtain a seed value for the $k+1_{th}$ group hash table
                                            ($seed_s^{k+1} = C \oplus E_{EK_s^k}(seed_s^k)$)
                                            Store the received tip value as a next hash value
                                            Generate a hash value for the next hash table
                                            Derive a group key for the next hash table and an encryption key
                                            ($KDF(K_s, ID_s, seed_s^{k+1}) = GK_s^{k+1} || EK_s^{k+1}$)
```

3.3 Attack-Resilient Algorithm based on Tree Structure

As we basically utilize only 18 bits for the hash value in the extended ID field of the CAN protocol, we cannot guarantee the collision probability is sufficiently low. The simplest way to defend against hash collision attacks is to store all of the possible hash values for authentication in a receiver ECU's memory. However, it is efficient for the receiver ECU not to keep all of the hash values because of the shortage of memory capacity. If a receiver ECU stores only one hash value at a time to authenticate each CAN message, it cannot authenticate the normal CAN message any more after the adversary succeeds in the hash collision attack on the receiver ECU. Therefore, a receiver ECU keeps at least two hash values for authentication. To make our SAP robust against the hash collision attacks coming from the third issue described in Section 2.3, we propose ART (Attack-Resilient algorithm based on the Tree structure). It basically constructs a certain tree having hash values as many as the height of the tree for a receiver ECU. The overall concept of ART is shown in Figure 5. For example, a sender ECU transmits the normal message with the *i+2*th authenticator from the hash chain and an attacker transmits attack messages masquerading the sender ECU. A receiver ECU keeps two hash values (the *i*th and the *i+1*th values) from the generated key chain in order to authenticate the normal sender ECU, where the height of the receiver ECU's tree is two. If the receiver ECU finds an apparent mismatch between the leaf's hash value and the hashed authenticator of the transmitted message, it should compare the hashed authenticator with the root's hash value. Otherwise, the value of the root is changed to the value of the leaf in the receiver ECU's tree and the value of leaf is replaced with this authenticator. If the hashed authenticator is the same as the hash value of the root, the tree extends a node as a new leaf to the root, which has this authenticator. If not, the message with this authenticator is dropped.

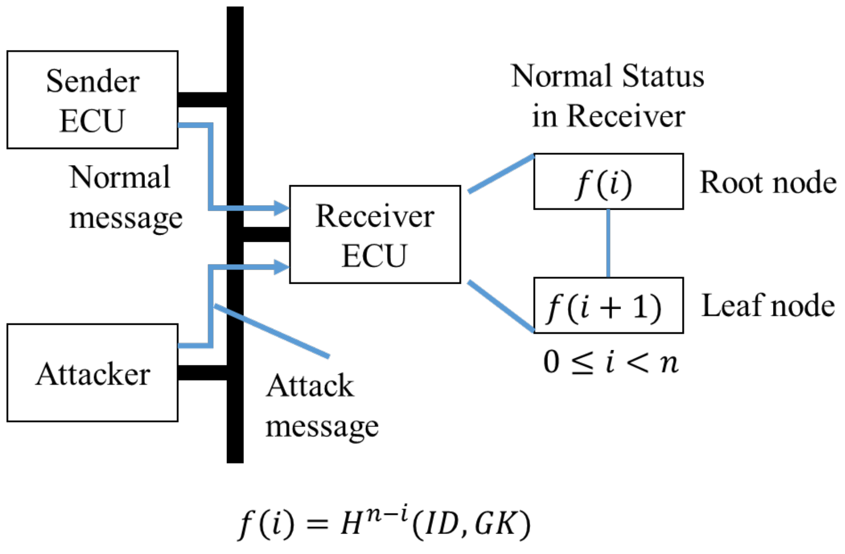


Figure 5. The general concept of ART (tree height = 2)

The pseudo code of ART is presented in Algorithm 1 where the height of the tree is *m* ($m \geq 2$). We can prevent a replay attack through filtering (from line 5 to 7 in Algorithm 1). The rest of our flow is for enduring the hash collision attack which is a kind of masquerade attack.

Algorithm 1 ART (Attack-Resilient Algorithm based on the Tree Structure)

```
 1: INPUT: inserted value $IN$, Hashed inserted value $H$, Tree list
        $TL$ (sorted in descending order in accordance with the level
        of tree and recentness)
 2: $N \leftarrow$ the number of nodes in tree
 3: for $i = 1$ to $N$ do
 4:         $COMP \leftarrow TL[i]$
 5:         if $COMP = IN$ then
 6:             Break
 7:         end if
 8:         if $COMP = H$ then
 9:             Create a new node
10:             New node parent $\leftarrow H$
11:             New node value $\leftarrow IN$
12:             Insert a new node to tree
13:             if $COMP =$ leaf node then
14:                 root node $\leftarrow$ ((Current level + 1) – $m$)th parent
15:                 Construct a new tree with the root node
16:                 (pruning the previous root node and other siblings of the new root
                    node)
17:             end if
18:             Authentication success
19:         end if
20: end for
21: Authentication fail (message drop)
```

IV. Security Analysis

4.1 Authentication

If an adversary knows how to generate the hash chain of SAP without ART, the probability of success in attacks is $O(2^9)$ when only 18 bits in the data frame of the CAN protocol is used as an authenticator. However, we do not consider such a scenario since it means that the adversary can physically access an ECU and know the firmware of it. As the adversary cannot obtain any information from the receiver ECU, they are not notified whether their hash collision attacks are succeeded or failed. Hence, they should repeatedly insert a random hash value in the extended ID field of the attack message. There is no sense in injecting attack messages into the CAN environment within the short period of time (below 1 ms). This is because they are easily detected by the IDS (Intrusion Detection System) [20]. Therefore, we assume that the adversary transmits attack messages at an interval of the minimum 1 ms.

We calculate the collision probabilities using the birthday paradox under our assumption. In SAP with ART, while each receiver ECU uses one hash value from the one-way hash chain, an adversary predicts what the hash value is. The adversary continually chooses the hash value used by hash collision attacks for each run independently in order to break the in-vehicle network system since success or failure of each attack in the receiver ECUs (or the receiver ECU) is not known. The only possible successful attack is that at least two of them match the predefined answer at the given time for n answers. Equation (1) indicates the probability of collisions by using the birthday paradox, where n is the number of tries (e.g., the number of people in birthday paradox) and d is the range of output (e.g., the number of days in a year, 365 in birthday paradox).

$$q(n,d) = 1 - \left(\frac{d-1}{d}\right)^{n} \tag{1}$$

The probability of a series of hash collisions from equation (1) when the tree height is equal to two is given by

$$\mathrm{r(n,d)} = \sum_{i=1}^{n-1} \left(\frac{d-1}{d}\right)^{i-1} \cdot \left(\frac{1}{d}\right) \cdot q(n-i) \tag{2}$$

In hash collision attacks, there are two possible attacks of the adversary to the root node or to the leaf node, respectively. In order to succeed the attack to the root node, an adversary causes two hash collisions to the receiver ECU before one normal message is inserted in the CAN bus and is received by the received ECU. In addition to the attack to the root node, the attack to the leaf node is also considered.

In the attack to the leaf node, the adversary has to succeed two attacks before two normal messages are inserted. Hence, the probability of its attack success is specified by equation (3):

$$\mathrm{s(n,d)} = r(n,d) + r\left(\frac{n}{2}, d\right) \tag{3}$$

We analyze the probability of success in attacks injected with the transmission interval of 500 ms which is the longest message period among sender ECUs in sample signal sets for CAN [22, 25]. From the above equations, the probability of attack success is summarized in Table 3.

Table 3. The probability of attack success

The attack message period	The probability of 1 collision	The probability of 2 collisions
10ms	0.0001869%	0.00000008768%
1ms	0.19017%	0.0000090415%

4.2 Key Freshness

Sender ECUs generate a group key from a random value as a seed in order to construct hash chain. It is never regenerated using the same seed. Therefore, adversaries cannot predict a new group key even though they accidentally acquire a seed value of the previous hash chain.

4.3 Replay Attack

Although adversaries eavesdrop the CAN bus, it can never identify securely-shared seed values from transmitted messages. They also cannot generate the next hash values without the seed value. Because SAP is based on this OTP concept, it can thwart the replay attacks for which CAN is especially vulnerable.

V. Experimental Results

In this section, we evaluate SAP with ART by comparing it with the existing authentication protocol such as Keyed-Hashed Message Authentication Code (HMAC) based authentication protocol proposed by Samuel *et al*. [6]. The setting and the specifications of the tools for the experiments are shown in Figure 6 and Table 4. Table 5 indicates detail conditions for experiments.

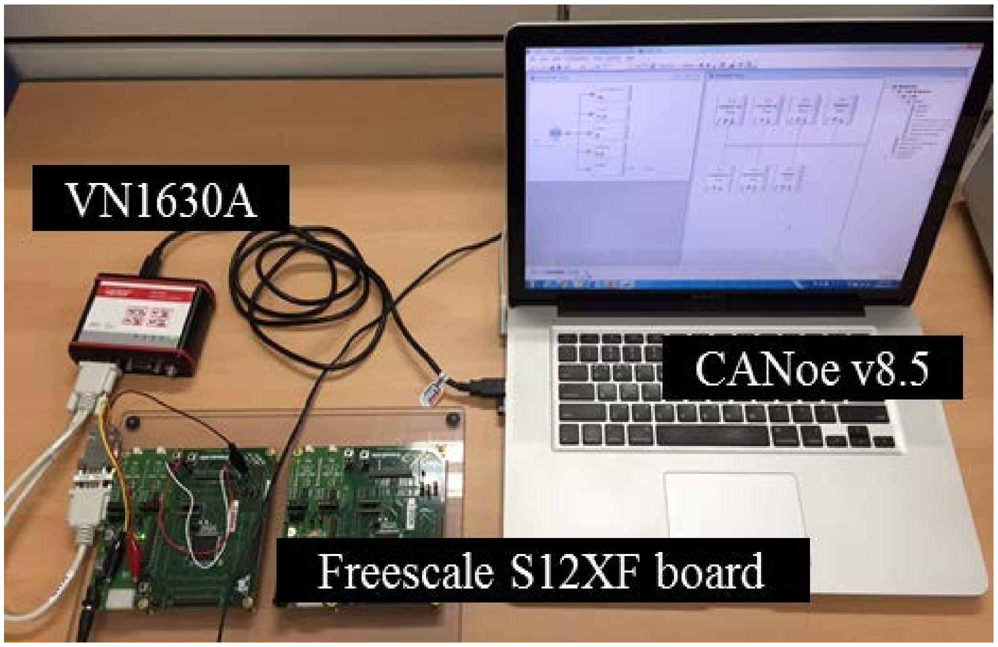


Figure 6. Experimental environment

Table 4. Specification of used tools

Tool	Model name	Note
Microcontroller	Freescale S12XF	40-60 MHz
Emulator	USB S08/HCS12 BDM Multilink	
Compiler	CodeWarrior	For Freescale MCU
SW	CANoe v8.5	Network simulator
Connector	VN1630A	Interface device

Table 5. Comparison conditions

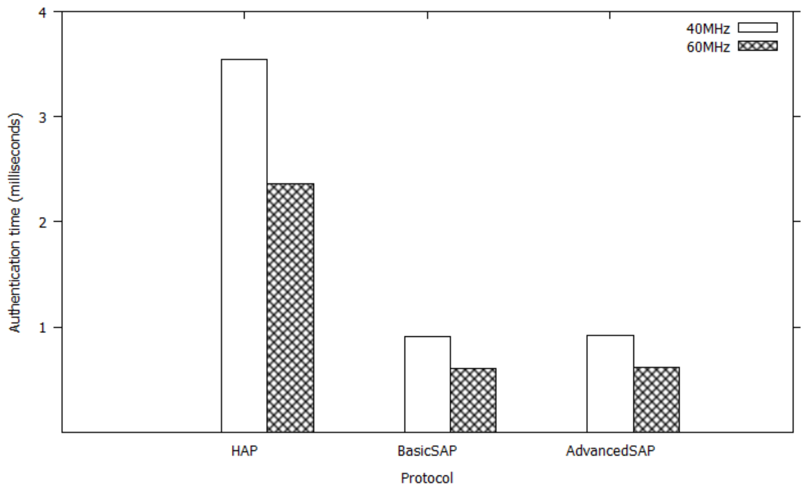
CAN bus speed	500 Kbps	Fixed
ECU data transmission cycle	10 milliseconds	Fixed
Number of ECU nodes	From 1 to 6 nodes	Variable

5.1 Hardware-Based Evaluation

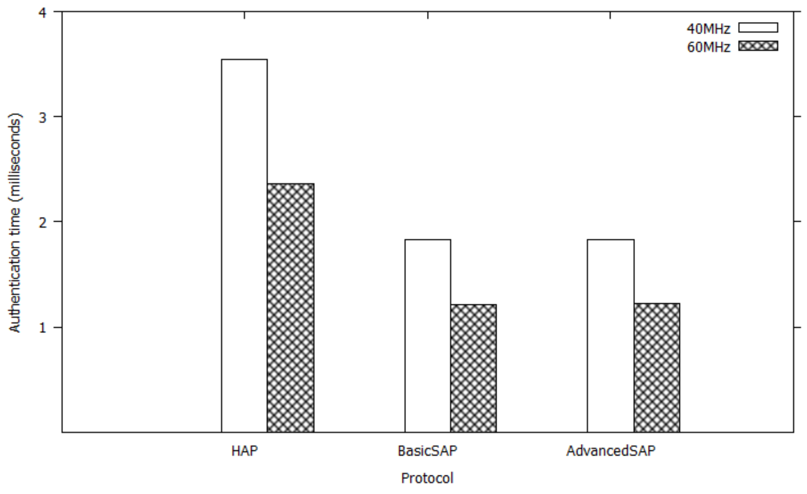
We implement Message Digest 5 (MD5), HMAC-MD5, and Advanced Encryption Standard-128 (AES-128) used for SAP and the authentication protocol in [6] on the S12XF microcontroller of Freescale. We adjust the clock rate of S12XF to 40 and 60MHz. In this evaluation, average execution time spent on both authentication and key update is measured during the execution of 10,000 times.

5.1.1 Authentication time

We define authentication time is average time to process authentication per one CAN message. The results are shown in Figure 7 where the *BasicSAP* is our authentication protocol SAP without ART and *AdvancedSAP* is SAP with ART. The results indicate that SAP is more efficient than HAP which is HMAC based Authentication Protocol [6]. The difference in authentication time between the sender ECU and the receiver ECU is due to the fact that the receiver ECU performs authentication of every received message by hash function. The results show that ART is lightweight because the overhead to authentication is only 4 microsecond in 60MHz clock rate.



(a) A sender ECU



(b) A receiver ECU

Figure 7. Authentication time

5.1.2 Key update time

The key update time represents average time for key update process per one update message. Note that the key update time is measured in a receiver ECU since HAP requires a gateway to generate and transmit a key update message. Figure 8 indicates the key update time in receiver ECU. The result shows SAP is almost 4 times faster than HAP. This is because AdvancedSAP shares only one message including the seed value while HAP requires two-way handshake for the key update. This difference comes from our one-way hash chain which can generate and share a seed value as a key before hash table exhausts all hash values.

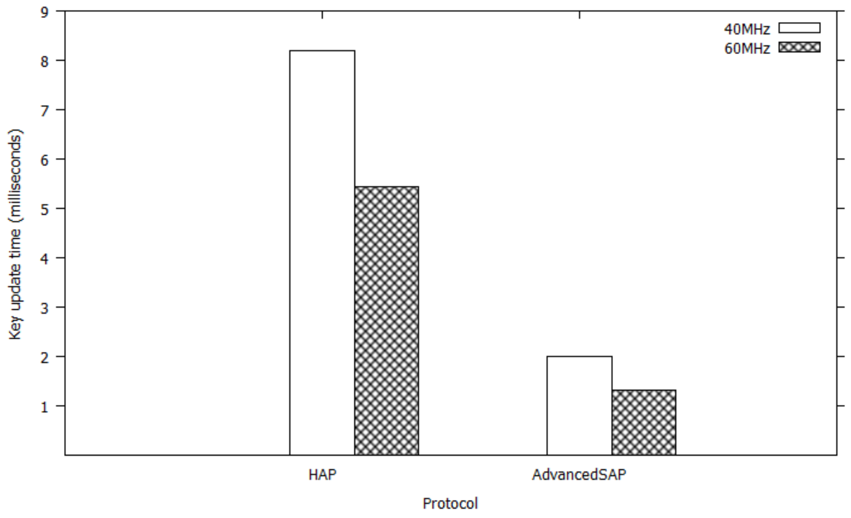


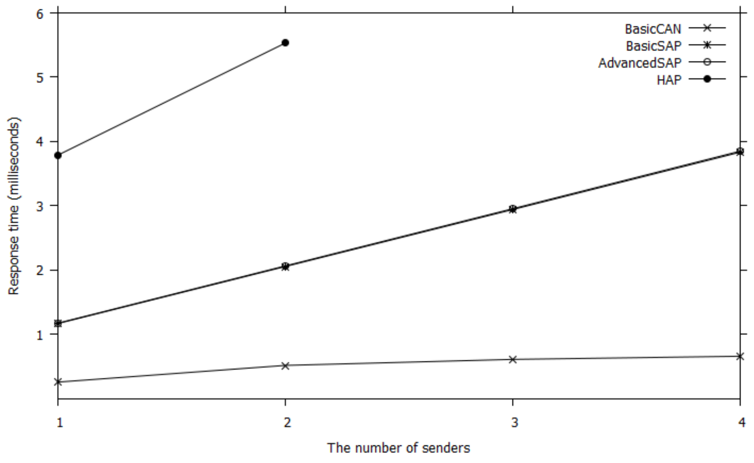
Figure 8. Key update time

5.2 Software-Hardware-Based Evaluation

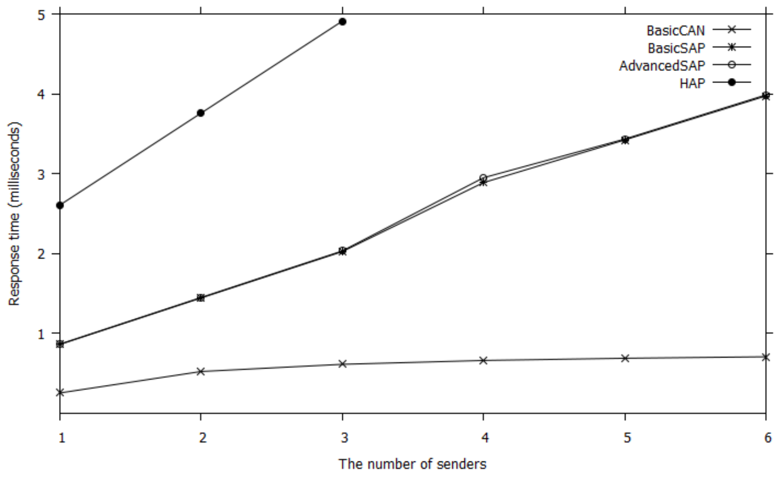
For this evaluation, we utilize S12XF and CANoe v8.5. CANoe is the network simulator for the vehicle by Vector Co [21]. The setting for this experiment is the same with Figure 6. There is only one real receiver ECU of S12XF. We create virtual sender ECU nodes in CANoe which transmit CAN messages. The number of virtual sender ECUs is adjusted from 1 to 6 nodes.

5.2.1 Response time

The response time is defined as average round-trip time which is the time required for sending a message from a virtual sender ECUs to the receiver ECU and back again. It is measured by adding the amount of time to send the message and to receive the response message in virtual sender ECUs, after the receiver ECU performs an authentication process and generates a response message. The results are shown in Figure 9 where BasicCAN performs the operation of general CAN without any security.



(a) 40MHz clock rate



(b) 60MHz clock rate

Figure 9. Response time

5.3 Software-Based Evaluation

For this evaluation, we implement both SAP and HAP on CANoe virtual ECU nodes and use Dynamic Linking Library (DLL). We construct a simulation environment based on an SAE benchmark CAN signal set [22, 28, 29]. It provides a sample in-vehicle network environment consisting of 7 ECUs and 53 signals with the deadline and the information of senders and receivers as shown in Figure 10. Table 6 indicates 17 messages formed by these signals and ID of CAN messages is arbitrarily determined. We utilize the authentication time and key update time measured in S12XF with 60MHZ clock rate as delay for virtual ECUs on CANoe. All message periods of this benchmark signal set are multiplied by 10 since S12XF is not capable of providing enough performance to conduct our experiments with the sampled signal set.

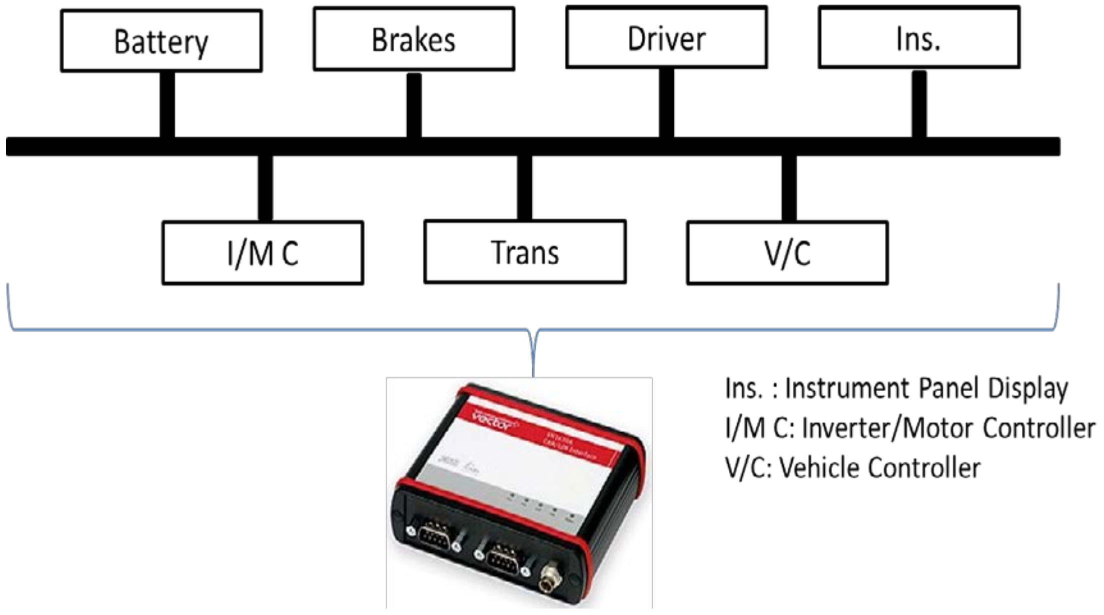

Battery
Brakes
Driver
Ins.
I/M C
Trans
V/C
Ins. : Instrument Panel Display
I/M C: Inverter/Motor Controller
V/C: Vehicle Controller

Figure 10. SAE benchmark in-vehicle network environment constructed in CANoe

Table 6. CAN messages constructed by SAE benchmark signal set

Message	Size (bytes)	Period (ms)	Deadline (ms)	Sender ECU
BAT1	2	5	5	Brakes
DRV1	1	5	5	Driver
IMC1	2	5	5	I/M C
TMC1	1	5	5	Trans
VHC1	2	5	5	V/C
VHC2	6	10	10	V/C
BAT1	1	10	10	Battery
DRV2	2	10	10	Driver
IMC2	2	10	10	I/M C
BRK2	1	100	20	Brakes
BAT2	4	100	100	Battery
BRK3	1	100	100	Brakes
TMC2	1	100	100	Trans
BAT3	1	1000	5	Battery
BAT4	3	1000	1000	Battery
TMC3	1	1000	1000	Trans
VHC3	1	1000	10000	V/C

5.3.1 Service delay

Transmission of messages in CAN occurs with an inter-arrival time generally termed the *period*. In other words, this period represents an interval at which transmission occurs. In CAN, the interval to a message is specified by the automobile manufacturers and a periodic message should be transmitted within the fixed interval (be known as deadline). Just before transmission of the periodic message, the processing of the message in ECU can occur with delay. It is the amount of time taken to process authentication, a key update, functions of CAN controller and transmitter, and competing for access to the bus. This delay is called *service delay*. The service delay of a given message *i* may be inherited from SAP and CAN, and is dependent on the ECU role (a sender or a receiver). Since a lightweight authentication protocol needs to minimize additional performance overhead incurred when ECU performs the authentication process, it is worth defining and measuring service delay as criteria to assess service availability of SAP. Service delay indicates how long the transmission of a sender ECU is delayed from the scheduled time at which the transmission to a given message must be started. Service delay should be less than the deadline of the given message. Otherwise, queued messages in ECU is subsequently dropped, and therefore, the authentication protocols based on the synchronization method might not work well. Note that service delay (d_s^i) to a given message i is measured by taking the difference between the actual time (t_1^i) and ideal time (t_0^i) that a transmission to message i is started. The following equations represent the service delay, average service delay and maximum service delay of a message *i*:

$$d_s^i = t_1^i - t_0^i \quad (4)$$

$$d_s = \frac{\sum_{i=1}^{m}(t_1^i - t_0^i)}{m}, \text{ where } m \text{ is the total number of transmitted messages} \quad (5)$$

$$max\ d_s^i = \{\max t_j^i \,|t_j^i = t_1^i - t_0^i,\ 0 < j \le n, n\ is\ the\ total\ number\ of\ i\ messages\ \} \quad (6)$$

From this experiment using hash chains with 1,000 values for AdvancedSAP, we collect 43,500 raw data for 5 minutes. The period of session key update of HAP is chosen arbitrarily to reflect the effects of session key update. We set it for 15, 30, and 60 seconds, respectively. Note that BasicCAN has a little delay consisting of processing delay of both a transmitter and a controller and competition delay of CAN.

The result of the average service delay is shown in Figure 11. The result indicates that the authentication method of AdvancedSAP is more lightweight than that of HMAC because the overhead for one message in AdvancedSAP is very low. In this case, AdvancedSAP is almost 4 times faster than HAP.

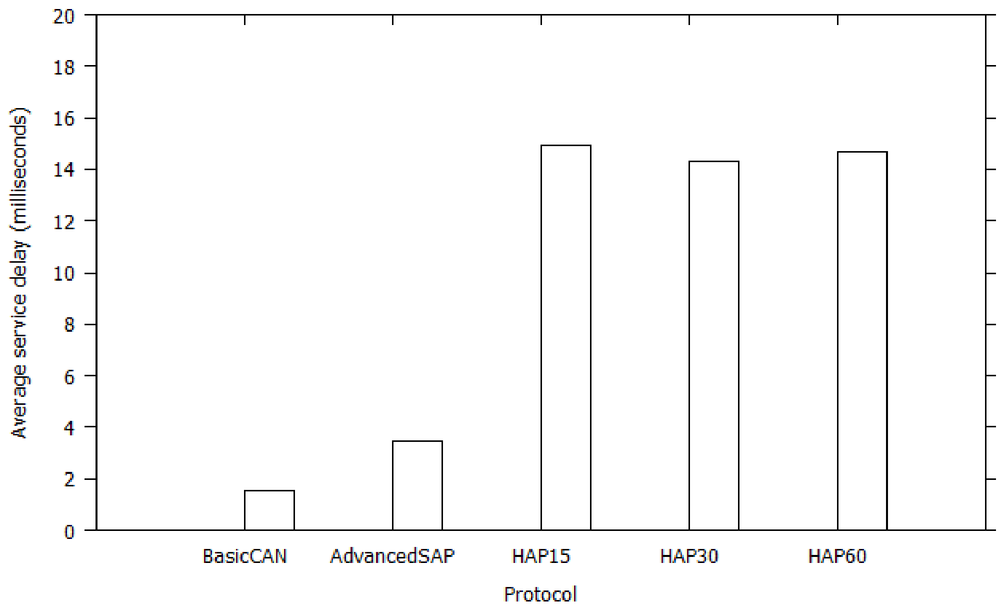


Figure 11. Average service delay on 60MHz clock rate

Figure 12 shows the maximum service delay as defined in equation (6). As above described, the deadline should not be longer than a specific period of each message [22]. As shown in Figure 12, HAP is not capable of finishing transmission within the deadline since the maximum service delay of the messages (such as BAT1, DRV1, IMC1, TMC1, and VHC1) with the period of 50 ms have almost 90 ms. On the other hand, since all of the messages by using AdvancedSAP are transmitted within the deadline, AdvancedSAP can be easily deployed in in-vehicle networks which consist of ECUs having much lower performance.

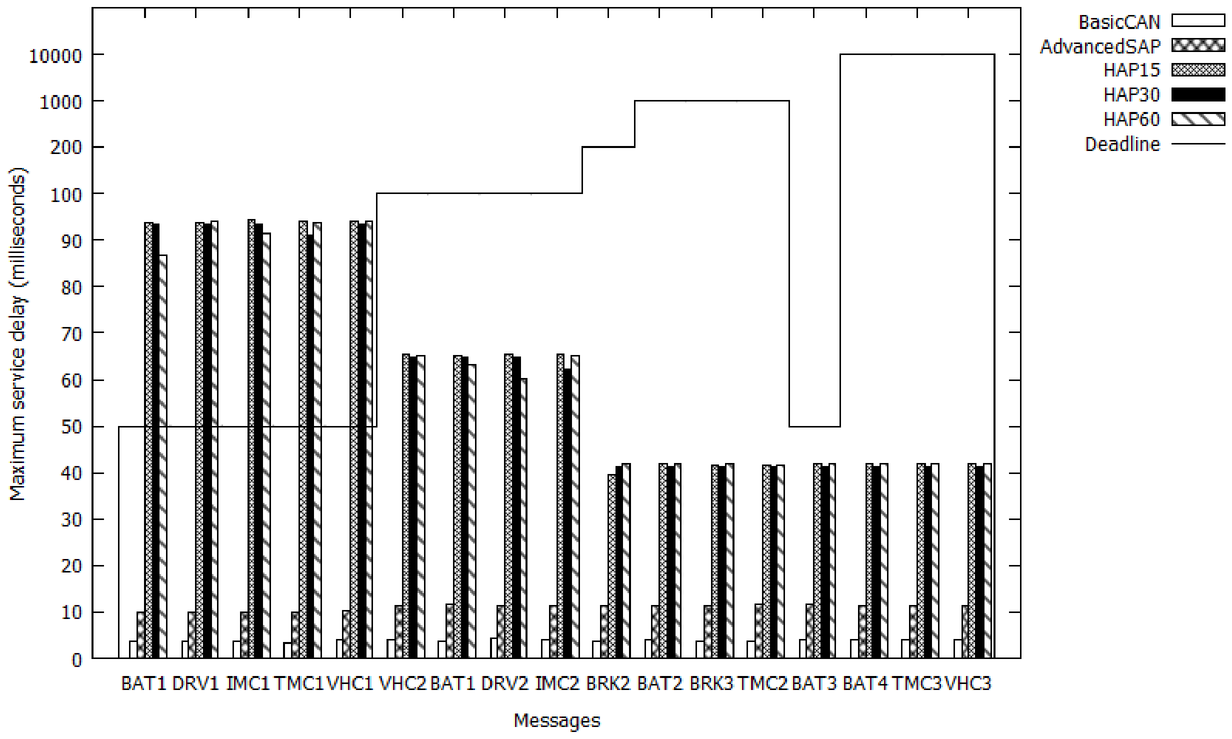


Figure 12. Maximum service delay per each message on 60MHz clock rate

5.4 Comparison Results

We have compared the performance of SAP with HAP in terms of authentication time, response time, and service delay. For authentication time at the receiver ECU, SAP is at least 2 times more efficient than HAP. For maximum service delay, SAP shows maximum of 8.5 times efficiency compared to HAP. Especially, it dramatically reduces the maximum delay of CAN related to deadline requirement in in-vehicle networks. This is because SAP carries out the proactive key management which shares the seed and generates the new hash values before exhausting all of the hash values from the hash table of the sender ECU. From the above results, we have shown that SAP is a lightweight authentication protocol which can be easily applied on CAN.

VI. Conclusions and Future Work

In this thesis, we propose a novel authentication protocol called SAP based on one-way hash chain with a sender-based group key. The proposed attack-resilient algorithm called ART thwarts hash collision attacks by adversaries. Using our authentication protocol, we can counteract masquerade attacks and replay attacks for which CAN is especially vulnerable. We have analytically shown the high security level of the proposed protocol and validate its effectiveness, compared with the existing authentication protocol. Our protocol is well suitable for the CAN environment since it not only provides lightweight authentication but also requires only firmware update to be deployed in CAN without any change of the CAN standard.

In the future, we plan to evaluate the performance of SAP in various commercialized chips and also develop more effective security solutions for in-vehicle networks.

.

References

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

[2] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, and T. Kohno. Comprehensive experimental analyses of automotive attack surfaces. In USENIX Security Symposium. San Francisco, 2011.

[3] R. Bosch. CAN specification version 2.0. Rober Bousch GmbH, Postfach, 300240, 1991.

[4] C.-W. Lin and A. Sangiovanni-Vincentelli. Cyber-security for the Controller Area Network (CAN) communication protocol. In Cyber Security (CyberSecurity), 2012 International Conference on, pages 1-7. IEEE, 2012.

[5] B. Groza, S. Murvay, A. Van Herrewege, and I. Verbauwhede. Libra-can: a lightweight broadcast authentication protocol for controller area networks. In International Conference on Cryptology and Network Security, pages 185-200. Springer, 2012.

[6] S. Woo, H. J. Jo, and D. H. Lee. A practical wireless attack on the connected car and security protocol for in-vehicle can. IEEE Transactions on Intelligent Transportation Systems, 16(2):993-1006, 2015.

[7] H. Schweppe, Y. Roudier, B. Weyl, L. Apvrille, and D. Scheuermann. Car2x communication: securing the last meter-a cost-effective approach for ensuring trust in car2x applications using in-vehicle symmetric cryptography. In Vehicular Technology Conference (VTC Fall), 2011 IEEE, pages 1-5. IEEE, 2011.

[8] B. Groza and S. Murvay. Efficient protocols for secure broadcast in controller area networks. IEEE Transactions on Industrial Informatics, 9(4):2034-2042, 2013.

[9] W. Voss. A comprehensible guide to controller area network. Copperhill Media, 2008.

[10] The EVITA project, 2008, Webpage. [Online]. Available: http://www.evita-project.org

[11] S. Woo, H. J. Jo, I. S. Kim, and D. H. Lee. A Practical Security Architecture for In-Vehicle CAN-FD. IEEE Transactions on Intelligent Transportation Systems 17.8: 2248-2261, 2016

[12] A. Perrig, R. Canetti, J. D. Tygar, and D. Song. The tesla broadcast authentication protocol. RSA CryptoBytes, 5, 2005.

[13] D. K. Nilsson, U. E. Larson, and E. Jonsson. Efficient in-vehicle delayed data authentication based on compound message authentication codes. In Vehicular Technology Conference, 2008. VTC 2008-Fall. IEEE 68th, pages 1-5. IEEE, 2008.

[14] C.-W. Lin, Q. Zhu, and A. Sangiovanni-Vincentelli. Security-aware modeling and efficient mapping for can-based real-time distributed automotive systems. IEEE Embedded Systems Letters, 7(1):11-14, 2015.

[15] N. Haller, C. Metz, P. Nesser, and M. Straw. A one-time password system. Technical report, 1998.

[16] N. Haller. The s/key one-time password system. 1995.

[17] K. Rhee, J. Kwak, S. Kim, and D. Won. Challenge-response based rfid authentication protocol for distributed database environment. In International Conference on Security in Pervasive Computing, pages 70-84. Springer, 2005.

[18] I. Syamsuddin, T. Dillon, E. Chang, and S. Han. A survey of rfid authentication protocols based on hash-chain method. In Convergence and Hybrid Information Technology, 2008. ICCIT'08. Third International Conference on, volume 2, pages 559-564. IEEE, 2008.

[19] L. Lamport. Password authentication with insecure communication. Communications of the ACM, 24(11):770-772, 1981.

[20] T. Hoppe, S. Kiltz, and J. Dittmann. Security threats to automotive can networks—practical examples and selected short-term countermeasures. Reliability Engineering & System Safety, 96(1):11-25, 2011.

[21] Vector, Webpage. [Online]. Available: www.vector-infomatik.com

[22] K. Tindell, A. Burns, and A. J. Wellings. Calculating controller area network (can) message response times. Control Engineering Practice, 3(8):1163-1169, 1995.

[23] M. Shavit, A. Gryc, and R. Miucic, “Firmware Update over the Air (FOTA) for Automotive Industry,” in Asia Pacific Automotive Engineering Conference, Hollywood, CA, USA, 2007.

[24] D. K. Nilsson and U. E. Larson. Conducting forensic investigations of cyber attacks on automobile in-vehicle networks. In Proceedings of the 1st international conference on Forensic applications and techniques in telecommunications, information, and multimedia and workshop, page 8. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[25] NetcarBench 3. 4 [Online], available: http://www.netcarbench.org/, August 1, 2014.

[26] R. Kurachi, Y. Matsubara, H. Takada, N. Adachi, Y. Miyashita, and S. Horihata. CaCAN-centralized authentication system in can (controller area network). In 14th Int. Conf. on Embedded Security in Cars (ESCAR 2014), 2014.

[27] B. Groza and T.-L. Dragomir. On the use of one-way chain based authentication protocols in secure control systems. In Availability, Reliability and Security, 2007. ARES 2007. The Second International Conference on, pages 1214-1221. IEEE, 2007.

[28] C. Class. Application requirement considerations.SAERecommended Practice J, 2056, 1993.

[29] K. Tindell and A. Burns. Guaranteed messagelatencies for distributed safety-critical hard real-timecontrol networks.Dept. of Computer Science,University of York, 1994.

요 약 문

CAN 에서 구현 가능한 경량의 단방향 해시 체인을 사용하는 송신자 인증 프로토콜

차량간 통신(V2X)의 발달로 인해 더 나은 연결된 서비스를 제공할 수 있게 되면서, 운전자의 안전과 편의성을 더욱 높일 수 있게 된 반면, 그런 서비스들은 차량에 대한 공격 범위 또한 함께 증가시켜왔다. 해커를 포함한 많은 연구자들이 이미 Controller Area Network (CAN) 으로 대표되는 차량 내부 네트워크의 취약점을 이용해 원격에서 차량을 제어하는 데모를 공개하였다. 본 논문에서 우리는 CAN 의 보안을 위해서 CAN 통신망을 구성하는 Electronic Control Unit (ECU) 들 간의 인증을 실시간성을 보장하는 범위에서 어떻게 수행할 수 있는지를 분석하고, 공격에 강인한 트리 알고리즘에 기반한 단방향 해시 체인을 사용하는 새로운 경량의 인증 프로토콜을 제안한다. 본 프로토콜은 펌웨어 업데이트만으로 실제 CAN 환경에 배치 가능하며, 높은 수준의 보안이 가능함을 보여준다. 또한, 그 성능은 가상의 ECU 들을 만들기 위한 CANoe 시뮬레이터와 실제 차량의 ECU 로서의 Freescale S12XF 를 사용하여 평가된다. 그 결과는 제안된 프로토콜이 기존의 인증 프로토콜들보다 인증 시간, 응답 시간, 그리고 서비스 지연 시간 측면에서 더 효율적임을 보여준다.

핵심어: 제어기 영역 네트워크, 차량 내부 네트워크 보안, 인증, 가상 물리 시스템