A Practical Wireless Attack on the Connected Car and Security Protocol for In-Vehicle CAN

Samuel Woo, Hyo Jin Jo, and Dong Hoon Lee, *Fellow, IEEE*

***Abstract*—Vehicle-IT convergence technology is a rapidly rising paradigm of modern vehicles, in which an electronic control unit (ECU) is used to control the vehicle electrical systems, and the controller area network (CAN), an in-vehicle network, is commonly used to construct an efficient network of ECUs. Unfortunately, security issues have not been treated properly in CAN, although CAN control messages could be life-critical. With the appearance of the connected car environment, in-vehicle networks (e.g., CAN) are now connected to external networks (e.g., 3G/4G mobile networks), enabling an adversary to perform a long-range wireless attack using CAN vulnerabilities. In this paper we show that a long-range wireless attack is physically possible using a real vehicle and malicious smartphone application in a connected car environment. We also propose a security protocol for CAN as a countermeasure designed in accordance with current CAN specifications. We evaluate the feasibility of the proposed security protocol using CANoe software and a DSP-F28335 microcontroller. Our results show that the proposed security protocol is more efficient than existing security protocols with respect to authentication delay and communication load.**

***Index Terms*—Connected car, controller area network (CAN), in-vehicle network security, key management.**

I. Introduction

THE newest model vehicles pursue convergence with various IT technologies to provide users with a comfortable driving environment and to effectively respond to auto emission regulations [1], [2]. In order to apply IT technology to vehicles, it is necessary to use a number of automotive application components. Among these components, the electronic control unit (ECU) is the most essential component that controls one or more of the electrical systems and subsystems in a vehicle [3]. State-of-the art vehicular on-board architectures can consist of more than 70 ECUs [4] that are interconnected via heterogeneous communication networks such as the controller area network (CAN), local interconnect network (LIN), or FlexRay [5].

As the most representative in-vehicle network, CAN has become the de facto standard because it dramatically decreases the number of communication lines required and ensures higher data transmission reliability [6]. Unfortunately, information security has not been considered in the design of CAN, although every bit of information transmitted could be critical to driver safety. For example, when data are broadcast using the BUS network, CAN does not ensure the confidentiality and authentication of the CAN data frame, paving the way for a malicious adversary to easily eavesdrop on data or launch a replay attack [7], [8]. The situation becomes worse when a vehicle is connected to automotive diagnostic tools. To check the functions of the ECUs during a diagnostic process, the tools broadcast CAN data frames without encryption and authentication to force control of the ECUs. This means that an adversary can also use an automotive diagnostic tool to easily get CAN data frames that can control an ECU [9].

Studies on vehicular security have been actively conducted mainly by European-funded projects (e.g., SEVECOM, PRECIOSA, EVITA, and OVERSEE) for the last ten years. The E-safety Vehicle Intrusion proTected Applications (EVITA) project specifically defined security requirements and developed appropriate solutions for vehicular on-board networks [10]. Among these solutions, EVITA-MEDIUM-HSM was developed in order to implement a secure communication environment among ECUs [11], [12]. However, EVITA does not provide a specific security architecture for a particular communication protocol. We note that a security technique used for the general IT environment cannot be immediately applied to CAN, as it has unique features such as a limited data payload. Therefore, it is necessary to design an efficient security technique even when EVITA-MEDIUM-HSM is used.

Security protocols in [13]–[15] were designed considering the limited data payload of the CAN data frame. However, these protocols are not suitable for deployment in the vehicle environment since they do not support real-time data processing and do not consider connection with external devices such as an automotive diagnostic tool. Considering the security vulnerabilities of aforementioned CAN, the appearance of the connected car that connects in-vehicle CAN to external networks enables wireless vehicle attacks [16].

In this paper, we demonstrate a practical wireless attack using a real vehicle in a connected car environment, in which a driver's smartphone is connected to the in-vehicle CAN. Our attack experiment consists of two phases: preliminary and actual attack. In the preliminary phase, i.e., before launching an actual attack, an attacker first acquires a CAN data frame to force control of the target vehicle using a diagnostic tool. In fact, the same model vehicles (more precisely, vehicles with the same configuration of automotive electronic subsystems) could be used. We note that a diagnostic tool is used to get a CAN data frame to force control of an ECU and does not need to be

Manuscript received January 13, 2014; revised May 2, 2014 and July 10, 2014; accepted August 8, 2014. Date of publication September 8, 2014; date of current version March 27, 2015. This work was supported by the Next-Generation Information Computing Development Program through the National Research Foundation of Korea funded by the Ministry of Science, ICT and Future Planning under Grant 2010-0020726. The Associate Editor for this paper was M. Chowdhury.

The authors are with the Center for Information Security, Korea University, Seoul 136-701, Korea (e-mail: samuelwoo@korea.ac.kr; hyojinjo86@gmail.com; donghlee@korea.ac.kr).

Digital Object Identifier 10.1109/TITS.2014.2351612

attached to the target vehicle during an actual attack. The attacker also manufactures a malicious self-diagnostic app that masquerades as a normal one and uploads it onto application markets. By using a self-diagnostic application such as "Torque," "Car Gauge Pro," and an OBD2 scan tool such as "EML327," "PLX KiWi," a driver can monitor CAN status information even while driving. Once the driver of the target vehicle downloads the malicious self-diagnostic app, the smartphone is under the control of the attacker. In the actual attack phase, exploiting the infected smartphone of the driver, a long-range wireless attack is carried out. That is, the attacker can inject CAN data frames independently of his location through the smartphone if mobile communication such as 3G, 4G, or LTE is possible. Hence, the smartphone does not need to belong to the attacker or a mechanic. Our proposed attack model is only possible when the driver downloads the malicious self-diagnostic app. Even there are approximately 300 self-diagnostic apps in application markets, they are not the most popular apps in terms of amount of app downloads. This reduces the possibility of attacks based on our proposed attack model. However, our proposed attack model is still a realistic attack scenario and would be practical in the near future since Google and leading car makers are collaborating to bring Android OS in vehicles and connected car services such as mirror link are increasing rapidly. Our attack experiment is explained in detail in Section IV.

Along with the practical wireless attack experiment, we also propose a security protocol to remedy the vulnerabilities of CAN satisfying the requirements in the following.

- The data encryption and authentication techniques ensure real-time data processing in the in-vehicle CAN.
- The method using a message authentication code (MAC) considers the limited data payload of the CAN data frame.
- Key management techniques support secure connectivity between external device and the in-vehicle CAN.

In addition, we evaluate the security and performance of the proposed security protocol using a manufactured Secure-ECU, similar to a real ECU and a commonly used commercial software tool. The following are the main contributions of this paper.

1) Demonstration of a practical long-range wireless attack experiment using a malicious smartphone app in a connected car environment.
2) Design of a security protocol that can be implemented on an ECU, accommodating the limited resources available and the current CAN data frame format.
3) Analysis of the security and performance of the proposed security protocol using Secure-ECU and CANoe.

II. Background

A. CAN

The CAN is a high-integrity serial data communication technology developed in the early 1980s by Robert Bosch GmbH for efficient communication between automotive applications. CAN is a multimaster broadcast communication bus system based on sender ID that allows ECUs to communicate on a single or dual wire network with data rates up to 1 Mb/s. The CAN protocol allowed auto makers to reduce the complexity and cost of in-vehicle network wiring; hence, ISO established CAN as the international standard in 1993 [17].

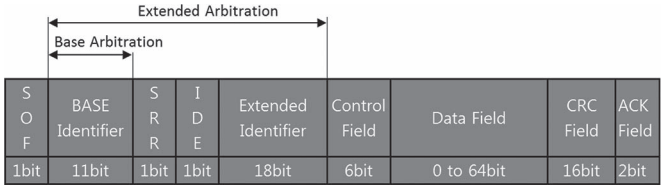

Fig. 1. Data frame format of CAN 2.0B protocol.

In the CAN protocol, each ECU transfers information to other ECUs using a data frame. A sender ECU transmits data frames that include its own ID. Other ECUs retrieve data frames selectively after identifying the ID of the sender ECU in the data frame. CAN protocol is divided into two modes, i.e., CAN 2.0A and CAN 2.0B, according to the length of the ID field. As CAN 2.0B supports compatibility with CAN 2.0A, we only describe CAN 2.0B in this paper. The data frame format of CAN 2.0B is shown in Fig. 1. CAN 2.0B has a 29-bit ID field divided into two parts: Base ID field and Extended ID field. The ID field is used to set the message priority. The IDE field determines the use of the 18-bit Extended ID field. The data field is a maximum of 8 bytes and includes information to be transmitted from the sender ECU to others. The cyclic redundancy check (CRC) field is used for error detection of the transferred data frame. The other fields are not related to our work and hence not explained.

B. Connected Car Environment

The connected car is receiving much attention as the next-generation Vehicle-IT convergence technology due to the rapid development of mobile communication technology and the expansion of the smart device and application services. Many auto manufactures have been independently developing connected car technologies such as OnStar of GM or Connected Drive of BMW. In addition, with the popularity of a Pay-as-You-Drive insurance, a variety of electronic devices are being sold that connect to the car's OBD2 (On-Board Diagnostics) port and can be used by smartphone applications. In general, a connected car is a vehicle that is always connected to external networks while driving. As in [16] and [18], the components of a connected car are as follows:

- a vehicle with ECUs and an in-vehicle network;
- a portal to provide the vehicle with various services;
- a communication link to connect the vehicle and portal.

Fig. 2 shows a connected car environment containing these components. In a vehicle, a number of ECUs are installed and connected within CAN. The portal may be divided into Web-based and smartphone app-based services. Recently, with the high performance and popularization of mobile communication technologies, more connected car environments are using smartphones. Various apps supporting the connected car environment are now sold in app markets such as Google Play and the Apple App Store (e.g., Send To Car, UVO Smart Control, and BMW ConnectedDrive).

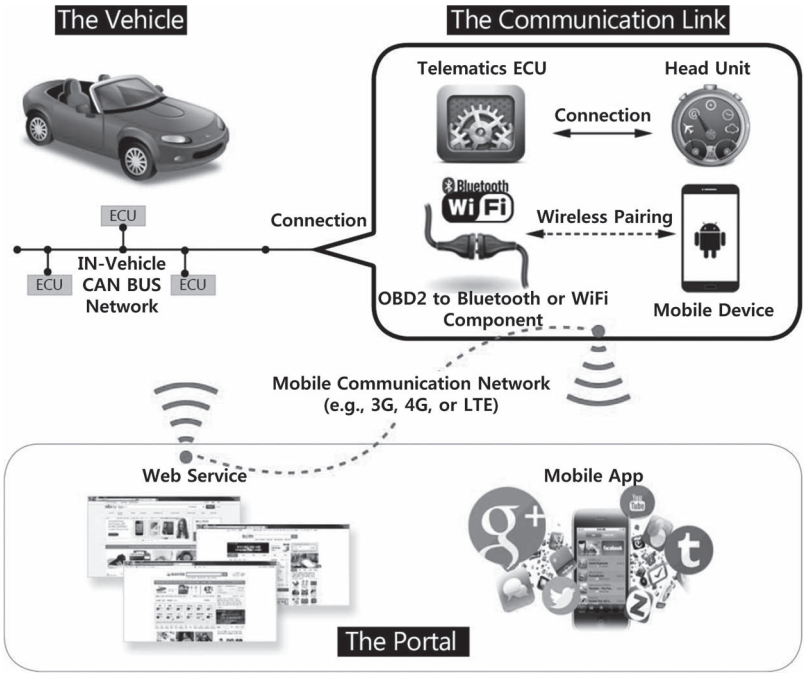

Fig. 2. Connected car environment.

TABLE I
CERT CLASSIFICATION OF THE PROPOSED ATTACK MODEL

Attacker	Tool	Vulnerability	Action	Target	Unauthorized result	Object
Hackers	Automotive diagnostic tool Malicious smartphone app	Design of CAN	Read Spoof	Vehicle with ECUs	ECU forced control	Challenge Thrill

III. ATTACK MODEL AND SECURITY REQUIREMENTS

Our attack model is the same as that of Koscher *et al.* [9], [19] in terms of exploiting the vulnerabilities of the in-vehicle CAN. However, the method of injecting malicious data into the in-vehicle CAN is clearly differentiated from those of the related work. Our attack model is designed based on an environment where a driver uses a self-diagnostic app to monitor status information after installing an OBD2 scan tool on the vehicle and then pairing it with his/her smartphone by Bluetooth. When the driver installs on his/her smartphone the malicious self-diagnostic app distributed by an attacker, the attacker can launch the actual attack. The attacker can obtain status information of the vehicle from the malicious self-diagnostic app and use it to inject malicious data into the in-vehicle network. Since the malicious self-diagnostic app and attacker's server communicate using the mobile communication network (e.g., 3G, 4G, or LTE), the attack is unconstrained by distance. Furthermore, as our attack model uses ECU forced control data commonly used for the same model (more precisely, vehicles with the same configuration of automotive electronic subsystems), it is not necessary to physically occupy the target vehicle in advance.

A. Attack Model

According to the Computer Emergency Response Team (CERT) taxonomy suggested by [8], we illustrate the classification of our proposed attack model in Table I. The assumptions for our attack model are as follows.

- ATTACKER ABILITIES: An adversary has access to an automotive diagnostic tool to acquire a CAN data frame to force control of an ECU before launching an actual attack. The attacker can eavesdrop and inject the CAN data frame using a malicious self-diagnostic app into the in-vehicle CAN in the connected car environment. Thus, the attacker does not have to attack the target from a short range. The app may be widely spread through the app markets by masquerading as a legitimate self-diagnostic app for a vehicle.
- TARGET VULNERABILITIES: The target vehicle uses CAN to communicate among ECUs. As mentioned in [3], [7], [8], and [9], CAN does not offer security services such as encryption or data frame authentication. This means that eavesdropping and replay attack in CAN are possible. The unauthorized use of automotive diagnostic tools is also a security hole since the tool stores control commands for the ECUs.
- VICTIM BEHAVIOR: The victim of the target vehicle downloads the malicious self-diagnostic app to his/her smartphone through an app market. The victim does not recognize that the app is performing malicious acts such as eavesdropping or replay attack on the in-vehicle CAN. In our proposed attack model, we do not consider an attack to compromise the ECU installed on the vehicle inside or an attack to manipulate the firmware of ECU, as these require a long period of occupancy of the target vehicle and specialized knowledge.

TABLE II
TOOLS USED FOR THE ATTACK EXPERIMENT

Product	Model Name	Used During	Functionality and Characteristics	Product Detail Info
Diagnostic Tool	HI-DS Scanner Gold	Preliminary Phase	•Generation of ECU forced control data frames The authors of [9] used a diagnostic tool for analyzing CAN data frame	•gitauto.com Commonly used tool in auto repair shop
CAN BUS Monitoring Tool	PCAN Explorer	Preliminary Phase	•CAN data frame monitoring and capture The authors of [9] manufactured CARSHARK tool for CAN bus monitoring	•peak-system.com Commonly used tool in CAN bus system
Vehicle	Midsize car	Preliminary Phase	•Analysis and acquisition of In-Vehicle CAN data •Response analysis to ECU control data	Omitted
		Actual Attack Phase	•Used as target vehicle The authors of [9] and [19] conducted a hacking experiment using the real vehicle	
Wireless OBD2 Scan Tool	CANlink Bluetooth	Actual Attack Phase	•Support of communication between In-Vehicle CAN BUS and the app installed on the driver's smartphone	•rmcan.com Wireless CAN interface
Smartphone	Samsung Galaxy S2	Actual Attack Phase	•Install of malicious self-diagnostic app The authors of [19] introduced a short-range wireless attack model using Smartphone	•samsung.com/us/ Based on Android OS
Attacker's Server	Desktop PC	Actual Attack Phase	•Communication with malicious self-diagnostic app	Intel i5-2500 Based on Win XP

TABLE III
AUTO MANUFACTURER AUTOMOTIVE DIAGNOSTIC TOOLS

Company	Tool Name	Company	Tool Name
Audi/Volkswagen	VAS5052	Chrysler	StarScan
Ford	FoCOM	Peugeot/Citroen	XS Evolution Multiplexer
Benz	Benz XP-Star Diagnosis	HONDA	HDS
BMW	BMW ICOM A+B+C	Toyota/Lexus	Intelligent TesterII
Hyundai/KIA	HI-DS Scanner Gold	Nissan	Consult

B. Security Requirements

Previous work has illustrated various types of attacks possible on vehicles. We note that all these attacks eventually stem from the vulnerabilities of in-vehicle CAN, as discussed in Section II. There are three main vulnerabilities of in-vehicle CAN: 1) weak access control; 2) no encryption; and 3) no authentication. In order to construct a secure in-vehicle CAN, these three vulnerabilities have to be eliminated. However, an in-vehicle CAN is a communication environment where access control is virtually impossible. As an in-vehicle CAN is a multimaster broadcast communication environment based on the sender's ID, a connected node may receive any data frame transmitted. In addition, a malicious node may transmit a data frame by stealing the ID of a normal node (e.g., a modification and replay attack of a data frame). Since in-vehicle CAN access control is virtually impossible by the nature of the broadcast communication, it is necessary to encrypt and authenticate data frames to prevent modification and replay attack. We identify the requirements to provide a secure in-vehicle CAN as follows.

• CONFIDENTIALITY: Every data frame in CAN should be encrypted to provide confidentiality. That is, the plain text form of the data frame should be only available to a legitimate ECU or party. Due to the nature of CAN, all the data frames are broadcasted, enabling an attacker to easily eavesdrop on a CAN data frame. In particular, a data frame to force ECU control can be obtained using an automotive diagnostic tool, hence it is very easy to analyze the meaning of a relevant data frame.

• AUTHENTICATION: A control data frame in CAN is identified only by the sender ID in the data frame, which makes a replay attack possible. That is, an adversary with a valid control data frame can retransmit it, possibly masquerading as a legitimate sender. To thwart this type of attack, both the authentication and integrity of the transmitted data should be provided. The current CAN specification only offers a CRC code to ensure transmission error detection, not authentication.

IV. PRACTICAL ATTACK EXPERIMENT

Based on the proposed attack model, this section describes a long-range wireless attack scenario and gives the results of our attack experiment. Our practical wireless attack experiment is carried out in two phases: preliminary and actual attack. The tools used in the experiment are listed in Table II.

A. Preliminary Phase

• *Use of an automotive diagnostic tool to acquire CAN data frames to force control of ECUs:* To identify the data frames controlling the critical components of the vehicle, Koscher *et al.* suggested various techniques in [9]. Among them, exploiting an automotive diagnostic tool is very simple because the tool stores control commands for the ECUs. Global auto manufacturers offer diverse kinds of automotive diagnostic tools for convenient diagnosis. Table III shows various automotive diagnostic tools. Our acquisition process using an automotive diagnostic tool is as follows.

1) An automotive diagnostic tool is connected to the OBD2 port of a vehicle, as shown in Fig. 3(a).

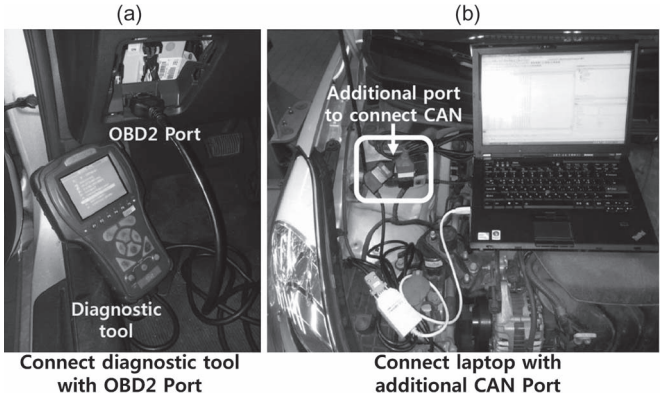

Fig. 3. Experimental environment to analyze the CAN data frame.

2) The in-vehicle CAN buses are monitored after connecting a laptop to an additional port, as shown in Fig. 3(b).
3) A command is performed to forcibly actuate a certain ECU using the automotive diagnostic tool.

When the aforementioned three steps have been performed, the laptop connected to the additional port can obtain CAN data frames to force control of an ECU. We acquired various kinds of ECU controlling CAN data frames in the experiment. In particular, we obtained the CAN data frame controlling an injector to shut off fueling and were able to make the engine shut down. We also analyzed the CAN data frames generated during a vehicle's normal driving. We repeatedly analyzed the characteristics of data frames generated by reproducing typical driving states (e.g., rapid acceleration and execution of the Smart Parking Assist System). Table IV shows the CAN data frames to force control of ECUs analyzed in our experiment. (In order to prevent the malicious use of our research findings, the complete byte information of the analyzed data frames is omitted.) CAN data frames acquired during normal vehicle driving are shown in frames 1–5 and those acquired with the diagnostic tool are shown in frames 6 and 7. The in-vehicle ECUs uses IDs ranging from 0x000 to 0x5FF and the automotive diagnostic tool uses an ID ranging from 0x700 to 0x7FF. Because they use different ID ranges, it is possible to easily identify the CAN data frames to control ECUs that have been generated from the automotive diagnostic tool.

• *Production and distribution of a malicious smartphone app for attack:* As of December 2013, there are more than 300 different smartphone apps for vehicles being distributed on app markets. We produced a malicious Android app masquerading as a self-diagnostic app for vehicles. The malicious app shows vehicle speed and an ECU error code while transmitting the CAN data frame from an in-vehicle CAN to an attacker's server using a mobile communication network such as a 3G/4G network. In addition, it is possible to transmit an ECU controlling CAN data frame from the attacker's server to the in-vehicle CAN. To produce the malicious app, we installed JAVA JDK, Android ADT, and Android SDK on a Windows 7-based PC and developed the app using Eclipse. We confirmed that the developed app works normally while driving on a real smartphone. Fig. 4 shows the state diagram of the malicious self-diagnostic app we produced and screenshots of the malicious self-diagnostic app. We did not actually upload and distribute our malicious app on an app market; however, as shown in [19], it is easy to do so. However, there is a weak point in the attack model using a malicious self-diagnostic app. Once an attacker has used the app to control a vehicle, the reputation of the app will start to decrease because Android users will shortly notice that the app is malware. A future research issue to strengthen the attack model could be on concealing actions to avoid the attack being noticed by users.

B. Actual Attack Phase

• *ECU forced actuation attack through the malicious smartphone app:* Using an Android smartphone, a server, an OBD2 scan tool, and a midsize car, we organized an environment as shown in Fig. 5(a) and performed an experimental attack. Fig. 5(b) shows the steps of the proposed attack scenario. After the preliminary phase is completed, the actual attack phase is launched when a driver installs the malicious self-diagnostic app onto his/her smartphone and uses it. A diagnostic tool is not physically attached to the target vehicle during the attack. The OBD2 scan tool is installed on the target vehicle and paired with the driver's smartphone by Bluetooth. The malicious self-diagnostic app and attacker's server are then connected using a mobile communication network. The experiment was done as follows.

1) The malicious app was installed on the victim's smartphone.
2) The victim connected the smartphone to the target vehicle using Bluetooth or WiFi. The malicious app provided the victim with normal functions, masquerading as a self-diagnostic app.
3) The malicious app transmitted data frames of the in-vehicle CAN to the attacker's server using the smartphone's mobile communication network. The attacker's server checked the state of the target vehicle and transmitted a CAN data frame to force control of an ECU to the in-vehicle CAN via the malicious app.
4) The target vehicle had a physical malfunction caused by the abnormal control data that was transmitted from the attacker's server.

Our attack experiment was reported on a television news program. The news video clip is available at URL http://goo.gl/mo33ay. The text on the URL is translated into English and attached to the Appendix. The Appendix explains details of the experiment between 28 s $\sim$ 1 min 46 s in the news video clip. The news video shows four types of attack experiments: distortion of the dash board, engine stop, handle control, and acceleration. In the news video, we manufactured and used the attacker app only for broadcasting. The attacker app, running on the attacker's smartphone, sends an attack message (i.e., the CAN data frame to force control of an ECU) to the attacker's server that then transmits and injects the attack message into the in-vehicle CAN of the target vehicle through the driver's infected smartphone. In our attack model, the attacker's server directly injects the attack message without using the attacker's smartphone (the attacker app).

TABLE IV
CAN DATA FRAME FOR FORCED CONTROL OF ECU

	CAN ID	Data Frame (8 Byte)	Vehicle's State-Transition in Case of Data Frame Replay-Attack	Analytic Method	Data Frame Transmission Site
1	0x43F	16 48 · · · · 00 F0	Rapid acceleration (max 200 km/h)	Data frame sniffing and replay during driving	Installed ECU in vehicle
2	0x440	FF 4E · · · · 09 00	Speedometer info change		
		16 48 · · · · 09 00	Fix speedometer to 0 km/h		
3	0x505	03 03 · · · · 00 00 04 03 · · · · 00 00	Dash board info change (SPAS-related)		
		IE 03 · · · · 00 00	SPAS malfunction		
		0A 03 · · · · 00 00 14 03 · · · · 00 00	Voice notification play (SPAS-related)		
4	0x545	FF FF · · · · FF FF	Dash board info change (fuel residual quantity)		
5	0x59B	00 05 · · · · 00 00 00 07 · · · · 00 00	Dash board info change (gear shift)		
6	0x7DF	02 10 · · · · 00 00	Imitate diagnostic tool connection	Data frame monitoring during diagnostic tool execution	Diagnostic tool
7	0x7E0	01 20 · · · · 00 00			
		04 30 49 · · · · 00	No. 1 Injector ON/OFF (Engine shut down)		
		04 30 4A · · · · 00	No. 2 Injector ON/OFF (Engine shut down)		
		04 30 4B · · · · 00	No. 3 Injector ON/OFF (Engine shut down)		
		04 30 4C · · · · 00	No. 4 Injector ON/OFF (Engine shut down)		

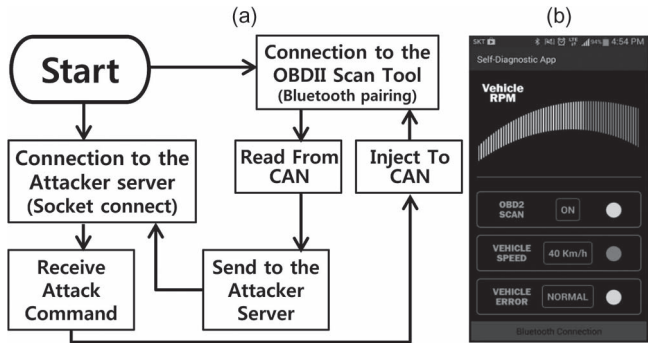

Fig. 4. State diagram of the malicious self-diagnostic app and screenshot of the malicious self-diagnostic app.

V. DEFENSE

A. Design Goal

To design a secure and efficient security protocol given the low-performance of an ECU and the limited data payload of a CAN data frame, these goals should be achieved.

• ENCRYPTION AND AUTHENTICATION FOR CAN DATA FRAMES: To provide confidentiality for a broadcasted data frame, it should be transmitted after encryption. In addition, to authenticate a transmitted data frame, a MAC should be generated and transmitted along with it. However, it is not easy to efficiently include a MAC in CAN data frame. As shown in Fig. 1, the CAN data frame consists of 120 bits, including a 64-bit data field. If the data field is used for a MAC, the total amount of CAN data frame transmission increases at least twice: one data frame for the original data and at least one for a MAC. Such a method is not proper since it will increase the CAN bus load rapidly. Therefore, a security protocol needs an efficient data authentication technique that can be applied to the current CAN data frame format.

• AN EFFICIENT KEY MANAGEMENT FOR CAN: For secure communication, a CAN security protocol should offer a secure and fast key distribution mechanism for data frame encryption and authentication. In addition, an efficient and secure session key update protocol is needed in order to enhance the security of the session key and truncated MAC. It is also needed to support the connectivity between an external device and a vehicle. For secure session key updates, two properties should be ensured:

1) FORWARD AND BACKWARD SECRECY: In CAN, every ECU should use an authentication session key AK_n and an encryption session key EK_n to ensure the authentication and confidentiality of CAN data frames in the n_{th} session. Keys AK_n and EK_n should not be used for decryption and authentication of the CAN data frames broadcasted in the $n+1_{th}$ session and $n-1_{th}$ session (forward and backward secrecy). If forward and backward secrecy cannot be ensured, regardless of the key update cycle, confidentiality, and authentication of the CAN data frames cannot be guaranteed.
2) KEY FRESHNESS: The encryption key EK_n and authentication key AK_n, which are used for encryption and authentication of CAN data frames in the n_{th} session, should be freshly generated. In order to ensure key freshness, a parameter used for key generation should be continually changed using a random number or counter. Key freshness is essential for preventing a replay attack [20].

B. Proposed Security Protocol

In order to describe our proposed security protocol, we assume the following. First, the gateway and general ECUs preshare the long-term symmetric keys K and GK. Second, the loading of long-term symmetric keys is done through a secure channel. Third, ECUs use the message filtering functionality of CAN. Each ECU registers the ID of sender-ECU on its ID table, and each ECU receives a packet only if it has the sender-ID registered on its ID table. For other packets, it performs filtering. Fourth, the sender and receiver ECUs synchronize data frames with a counter. When a data frame is completely transferred to the receiver, both increment a data frame counter. (In CAN, senders and receivers reciprocally check the transmission

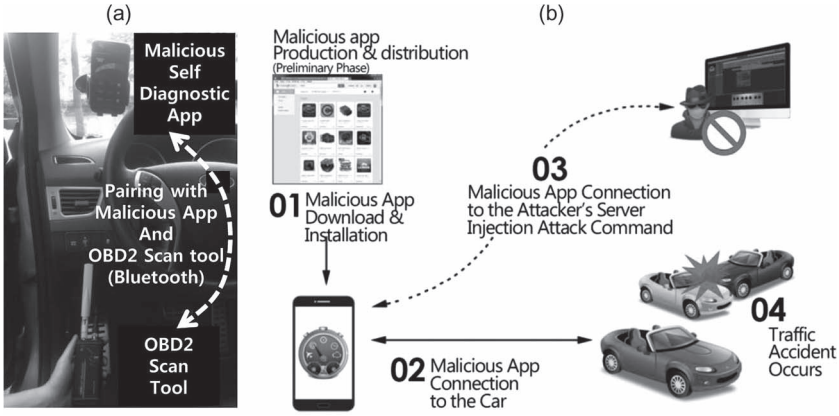

Fig. 5. Practical wireless attack experiment environment and steps of the proposed attack scenario.

TABLE V
NOTATION USED FOR PROPOSED MECHANISM

Notation	Description
ECU_i	ECU using identity i
ID_i	Identity of ECU_i
GECU	Gateway ECU
CTR_{ECU_i}	ECU_i data frame counter
$Seed_k$	Seed value of k_{th} session
EK_k	Encryption key of k_{th} session
AK_k	Authentication key of k_{th} session
C	Ciphertext
M	Plaintext
KEK_k	Key encryption key of k_{th} session (general key update phase)
KGK_k	Key generation key of k_{th} session (key generation key used for session key update process)
UK	Key encryption key of new session (when releasing an External Device Connection)
K_i	Long-term symmetric key between GECU and ECU_i (authentication key used for initial session key derivation)
GK	Long-term symmetric key between GECU and All ECU (key generation key used for initial session key derivation)
$KDF_x()$	Keyed one-way function used for key derivation
$H_{1,x}$	Keyed hash function using x $H_{1,x} : \{0,1\}^* \times key \rightarrow \{0,1\}^{64}$
$H_{2,x}$	Keyed hash function using x $H_{2,x} : \{0,1\}^* \times key \rightarrow \{0,1\}^{32}$

state of the CAN data frame using the ACK bit field. Hence, between the sender and receiver, it is possible to manage and synchronize a data frame counter.) Fifth, the gateway ECU has higher computing power than a general ECU. Sixth, a device certificate is loaded onto the gateway ECU and external devices.

Our proposed security protocol is divided into five phases. Phases 1 and 2 use a well-known security technique (Long-term symmetric key and Authenticated Key Exchange Protocol 2 (AKEP2)) in order to construct an initial session key distribution. In the security protocol we propose, the main novel contributions lie in Phases 3, 4, and 5. The notations used in this paper are listed in Table V.

1) Loading Long-Term Symmetric Keys: ECU_i loads long-term symmetric keys K_i and GK into secure storage. The GECU loads N of the K_i and a GK into secure storage. (The value N is the number of ECUs installed on the vehicle.) The long-term symmetric keys loading phase is performed only when manufacturing a vehicle or changing an ECU.

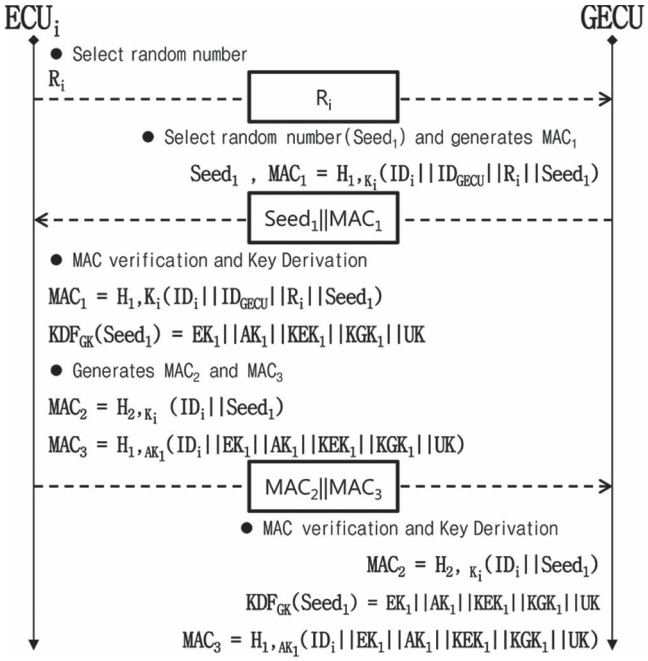

Fig. 6. Distribution of initial session key.

2) Distribution of Initial Session Keys: After starting a vehicle, every ECU performs an initial session key derivation process with GECU in a fixed order. While GECU derives the initial session keys with a particular ECU, other ECUs do not communicate but wait their turn. We used AKEP2 to construct a secure and efficient key derivation process in the in-vehicle CAN environment, as it provides mutual entity authentication and implicit key distribution [21]. As [21] explicitly allows for the removal of redundant parts from the protocol without impacting security, we removed the redundant parts and added a value for key confirmation. The distribution of the initial session keys is shown in Fig. 6.

(A) ECU_i selects random number R_i and transmits it to GECU.

(B) GECU selects random number $Seed_1$, generates MAC_1 for ID_i, ID_{GECU}, R_i, and $Seed_1$ using K_i, and transmits it to ECU_i with $Seed_1$.

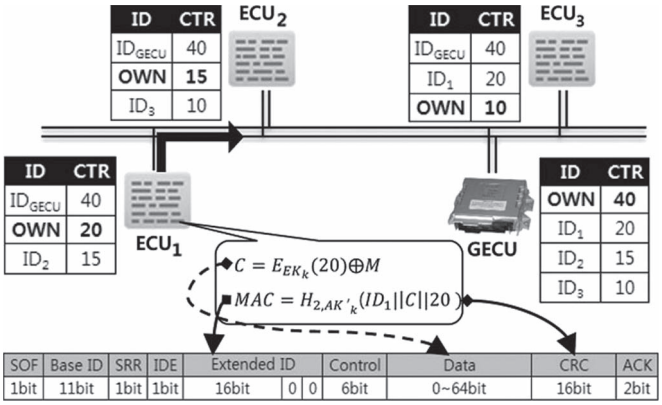

Fig. 7. Secure CAN data frame generation and counter table management.

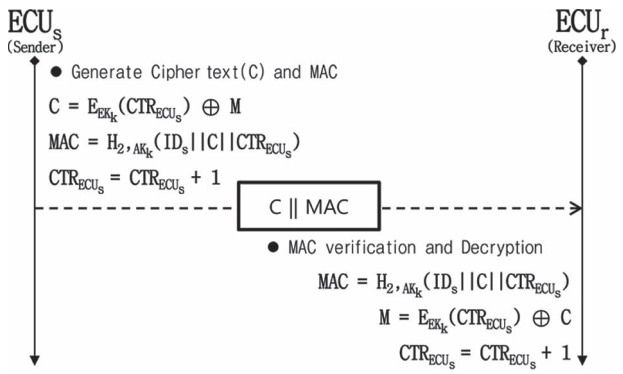

Fig. 8. Encryption and authentication of CAN data frame.

(C) ECU_i verifies MAC_1.

(D) ECU_i computes the initial session keys as

$$KDF_{GK}(Seed_1) = EK_1 \| AK_1 \| KEK_1 \| KGK_1 \| UK. \quad (1)$$

(E) ECU_i generates MAC_2 for ID_i and $Seed_1$ using K_i. It also generates MAC_3 for ID_i, EK_1, AK_1, KEK_1, KGK_1, and UK using AK_1. MAC_2 and MAC_3 are transmitted to GECU.

(F) GECU verifies MAC_2. By verifying MAC_2, it is possible to confirm that ECU_i received $Seed_1$ correctly. After MAC_2 verification, GECU computes the initial session keys as in (1).

(G) After generating the initial session keys, GECU verifies MAC_3 using AK_1. By verifying MAC_3, it is possible to confirm that ECU_i generated the initial session keys correctly.

3) Encryption and Authentication of CAN Data Frames: After the initial session key distribution process is complete, each ECU performs encryption and authentication of the data frames generated during a vehicle's normal driving. In our proposed protocol, we use the Advanced Encryption Standard-128 (AES-128) and the Keyed-Hash MAC. We propose two methods: Basic and Enhanced. CAN data frame encryption and authentication are shown in Figs. 7 and 8.

• **Basic Method**

Message Transmission (k_{th} **Session**)

(A) Sender ECU_s manages its own data frame counter value CTR_{ECU_s}. When transmitting a data frame, ECU_s generates ciphertext (C) using CTR_{ECU_s} from

$$C = E_{EK_k}(CTR_{ECU_s}) \oplus M. \quad (2)$$

When using the AES-128 algorithm, the result of $E_{EK_k}(CTR_{ECU_s})$ is 128-bit. As the maximum size of the CAN data payload is 64 bit, only the first 64 bits are used to generate ciphertext (C).

(B) ECU_s generates a MAC value for the CAN data frame that includes ciphertext (C) and CTR_{ECU_s} as follows:

$$MAC = H_{2,AK_k}(ID_s \| C \| CTR_{ECU_s}). \quad (3)$$

Since the data frame payload is 8 bytes, we use a truncated 32-bit MAC and a division method to transmit it. As shown in Fig. 1, the ID field of CAN 2.0B is separated into two subfields. Our proposed security protocol uses the first 16 bits in the extended ID field and the 16-bit CRC field for MAC transmission. In the 18-bit extended ID field, the unused two bits are set to zero. CRC is used for the integrity of the received data. Since MAC provides both data authentication and integrity simultaneously [22], it still ensures that the received data has not been altered. MAC verification delays may occur; however, the delay is not severe and does not interfere with the real-time processing of data. We experimentally show that MAC ensures a performance rate similar to that of a normal CAN. Section VII gives a detailed description of the performance evaluation experiment. Fig. 7 shows details of the division process. Once ECU_s completes the aforementioned steps, it transmits the secure CAN data frame and increments CTR_{ECU_s}.

Message Reception (k_{th} **Session**)

(A) Receiver ECU_r manages the counters of ECU_s. ECU_r verifies the MAC of the received secure CAN data frame using CTR_{ECU_s} and AK_k.

(B) When step A) is completed correctly, ECU_r performs decryption and obtains plaintext (M) using the following:

$$M = E_{EK_k}(CTR_{ECU_s}) \oplus C. \quad (4)$$

(C) After completing verification and decryption of the received data frame, ECU_r increments the CAN data frame counter of $ECU_s(CTR_{ECU_s})$.

• **Enhanced Method**

In the AES-128 algorithm, the result of $E_{EK_k}(CTR_{ECU_s})$ is always 128 bits. In the enhanced method, two CAN data frames are encrypted using the result of one AES encryption. Since the size of a CAN data payload is 64 bits, a single 128-bit calculation of $E_{EK_k}(CTR_{ECU_s})$ is used twice by dividing it into two parts (left and rightmost 64 bits). Thus, the computation cost for AES is half that of the basic method.

4) Key Update: A 32-bit truncated MAC is not enough to secure a CAN. In addition, it is possible for an adversary to leak a session key using an external device connection. Accordingly, the encryption and authentication keys used for each session should be updated periodically. In the k_{th} session, the proposed phases for key update are as follows.

• **General Key Update Phase** (k_{th} SESSION)

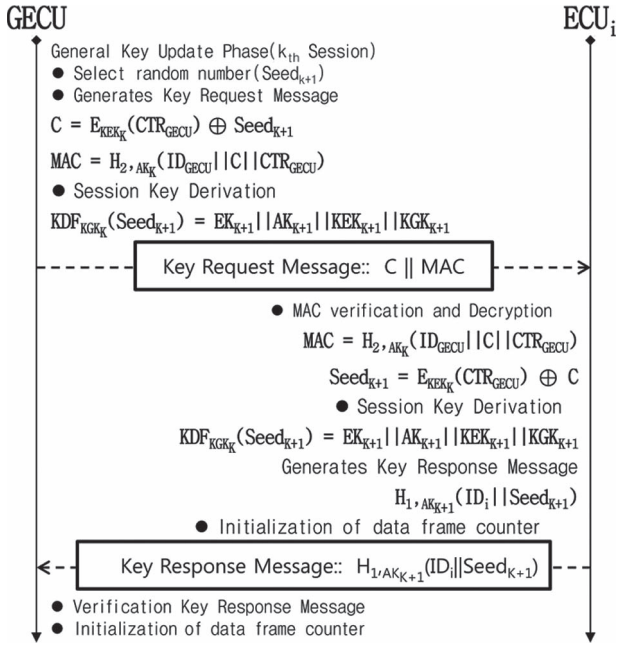

Fig. 9. Session key update process.

GECU performs the key update per predefined period (T) as follows. (Fig. 9 shows the key update process.)

(A) After selecting random values for Seed_{k+1}, GECU generates a Key Request Message and broadcasts it to the in-vehicle CAN. The message is formed of the following:

$$\mathsf{C} = \mathsf{E}_{\mathsf{KEK}_k}(\mathsf{CTR}_{\mathsf{GECU}}) \oplus \mathsf{Seed}_{k+1} \tag{5}$$

$$\mathsf{MAC} = \mathsf{H}_{2,\mathsf{AK}_k}(\mathsf{ID}_{\mathsf{GECU}}\|\mathsf{C}\|\mathsf{CTR}_{\mathsf{GECU}}). \tag{6}$$

(B) Every ECU_i that receives a Key Request Message performs verification of the MAC and decrypt ciphertext (C) using AK_k and KEK_k, respectively. Then, ECU_i derives session keys to be used in the $k+1_{th}$ session using a predefined function $\mathsf{KDF}()$ with KGK_k. Each data frame counter is also initialized to zero.

(C) After step B), each ECU_i generates a Key Response Message as follows and transmits it to GECU to confirm that they received the Key Request Message correctly:

$$\mathsf{H}_{1,AK_{k+1}}(\mathsf{ID}_i\|\mathsf{Seed}_{k+1}). \tag{7}$$

(D) After key confirmation, GECU initializes each data frame counter of ECU_i to zero. The key update phase is complete when GECU initializes its own data frame counter.

• Key Update Phase When Releasing an External Device Connection (k_{th} Session)

If a connection between an external device and a vehicle is terminated, GECU performs a session key update process to maintain secrecy as follows.

(A) GECU generates random values of Seed_{new}. Then it generates a Key Request Message using UK

$$\mathsf{C} = \mathsf{E}_{\mathsf{UK}}(\mathsf{CTR}_{\mathsf{GECU}}) \oplus \mathsf{Seed}_{\mathsf{new}} \tag{8}$$

$$\mathsf{MAC} = \mathsf{H}_{2,\mathsf{AK}_k}(\mathsf{ID}_{\mathsf{GECU}}\|\mathsf{C}\|\mathsf{CTR}_{\mathsf{GECU}}). \tag{9}$$

Then, GECU broadcasts the Key Request Message to the in-vehicle CAN.

(B) Every ECU_i that receives a Key Request Message performs verification of the MAC and decrypt ciphertext (C) using AK_k and UK, respectively. The rest of the steps are the same as the general key update phase.

5) Sharing a Session Key With an External Device: We propose a phase for additional authentication and key distribution when connecting a vehicle with external devices such as an automotive diagnostic tool, assuming it is a reliable device. In the k_{th} session, external device authentication and session key distribution are as follows.

(A) After connecting an external device to a vehicle, the external device sends an authentication request to GECU.

(B) After receiving the request, GECU generates a random number $\mathsf{r}_1(\mathsf{r}_1 \in Z_q^*)$, and a signature on $\mathsf{r}_1\mathrm{P}$. Then, it transmits these values to the external device with its certificate. (Where G is the subgroup of the elliptic curve group, P is a generator point of G and its order is a large prime q.)

(C) If the certificate and signature transmitted by GECU are successfully verified, the external device generates a random number $\mathsf{r}_2(\mathsf{r}_2 \in Z_q^*)$ and a signature on $\mathsf{r}_2\mathrm{P}$. Then, it transmits these values to GECU with its certificate. After completing transmission, the external device generates a temporal session key $(\mathsf{SK} = \mathsf{r}_1\mathsf{r}_2\mathrm{P})$ and removes r_2.

(D) After verifying the certificate and signature transmitted by the external device, GECU generates a temporal session key $(\mathsf{SK} = \mathsf{r}_1\mathsf{r}_2\mathrm{P})$ and removes r_1. Then, GECU encrypts Seed_k with the SK and transmits it to the external device. The external device derives the session keys to be used in the k_{th} session using Seed_k and is then able to communicate.

VI. Related Work

With the commercialization of the connected car, the in-vehicle CAN, which was regarded as a closed network in the past, is now being connected to external networks and provides useful services such as Remote Diagnostics [18], [23], Firmware Updates Over the Air [24], [25], and Real-time Product Carbon Footprints Data Analysis [26]. On the other hand, such connectivity to external networks introduces a new type of security threat to the vehicle.

Koscher *et al.* suggested specific wireless attack techniques and experimented with short- and long-range wireless attacks [19]. A short-range wireless attack is possible when a Bluetooth device installed on the vehicle is paired with the driver's smartphone on which a malicious app has been installed. A long-range wireless attack is possible owing to the vulnerability of the authentication function in the aqLink protocol. However, to conduct the wireless attacks in [19], complex and advanced technologies such as reverse engineering are required to analyze automotive electronics. In addition, the long-range wireless attack is possible only for a vehicle using the aqLink protocol. The previous studies on vehicular security point out vulnerabilities of the in-vehicle CAN as the primary cause of a

cyber attack [8], [13]. In particular, [9] mentions the lack of data frame authentication and encryption as the most severe vulnerabilities of CAN.

In order to construct a secure in-vehicle CAN, a variety of studies and research projects have been conducted over the past ten years. One of the European-funded projects, EVITA developed a hardware security module (HSM) for On-Board network security. HSMs may be classified into three types according to the field, in which they are used.

1) Full HSMs suitable for Vehicular Networks (Inter or Intra).
2) Medium HSMs suitable for Intra Vehicle Networks.
3) Light HSMs suitable for sensors and actuators.

Schweppe *et al.* suggested a communication security architecture for vehicles using EVITA-HSM in [11] and [12]. They used a truncated 32-bit MAC considering the limited data payload of CAN data frames and explained that a 32-bit MAC is secure from collision attacks for 35 weeks due to the limited properties of an in-vehicle network (CAN bus load and bandwidth). However, the security architecture of Schweppe *et al.* is very abstract. It does not provide a detailed description regarding how to generate and transmit a 32-bit MAC. It also does not consider data confidentiality and connectivity to external devices.

To provide an in-vehicle CAN communication environment secure against a replay attack, [14] and [15] proposed data authentication techniques that considered the limited data payload of a CAN data frame. Groza and Marvay suggested a CAN data authentication protocol using a TESLA-like protocol in [14]. In TESLA, a sender attaches to each data a MAC computed with a key k known only to the sender. A short time later, the sender sends k to the receiver, who can then authenticate the data. We note that key disclosure delay in the TESLA-like protocol should be minimized to ensure real-time processing in CAN. However, the shorter the delay is, the larger the bus load is. Our simulation in the next section shows that the TESLA-like protocol [14] finds it difficult to provide real-time processing in CAN. Groza *et al.* also proposed a single master case to minimize key disclosure delay. In a single master case, the sender generates a MAC with a long-term secret key shared with the communication master and transmits a data and the corresponding MAC to the master. The master then transmits the data and MAC to the receivers. However, since the secret key shared between a sender and the communication master cannot be changed for each session, a replay attack after eavesdropping on the transmitted CAN data frame and MAC is possible.

Lin *et al.* proposed a MAC generation technique using an ID table, message counter, and pair-wise symmetric key (PWSK) [15]. Receivers' IDs are registered on a sender's ID table. It is assumed that a sender shares a PWSK with the receivers in the ID table. Their MAC generation technique is similar to ours in that it uses a synchronized message counter among ECUs. However, the protocol of Lin *et al.* uses a PWSK, whereas ours uses a group session key. Using a PWSK implies that a sender must generate as many MACs as receivers in the communication group and transmit them separately to each receiver. This will increase the bus load rapidly and is hence impractical. In addition, their security technique does not consider data confidentiality and connectivity with external devices. In Section VII, we perform comparative evaluations among the proposed security protocol and those in [14] and [15].

VII. Security and Performance Analysis

A. Security Analysis

• Confidentiality: Our proposed security protocol uses AES encryption algorithms to ensure the confidentiality of the CAN data frame. When a vehicle is started, every ECU performs an initial session key derivation process with GECU using the long-term symmetric key $(\mathsf{K_i})$ and (GK). While the legitimate ECUs storing the long-term symmetric key $(\mathsf{K_i})$ and (GK) can compute the initial session key, an adversary cannot obtain it. This means that an adversary cannot acquire an encryption key (EK_1) and an authentication key (AK_1) derived from Seed_1. In subsequent sessions, the adversary cannot gain any session keys because $\mathsf{Seed_i}$ has been encrypted by KEK_{i-1}. Since the security of the AES algorithms has been proven in [27], it is clear that the adversary cannot obtain CAN data without a session key.

• Authentication: We use a 32-bit truncated MAC that is the same as the one used by EVITA-MEDIUM-HSM. According to [28], a 32-bit MAC can be broken after $O(2^{16})$ queries if an adversary can access the 32-bit MAC generation oracle. This means it is possible for an adversary, who can access the firmware of an ECU, to compromise it. However, we do not assume this type of adversary, as explained in Section III-A. It is also possible for the attacker to use the known structure of the input to a MAC to generate meaningful messages. However, the attacker still cannot generate the MAC corresponding to the meaningful message without knowing the MAC key. The key used to generate a MAC is securely shared in the proposed protocol. The only option for the attacker is to choose one 32-bit string out of 2^{32} possible MAC values. While a 32-bit MAC can be forged within a few seconds in a general IT environment that allows access to an MAC generation oracle, it takes about 11930 h for an adversary to transmit 2^{32} data frames per 10 ms for a forgery attack in a general in-vehicle CAN. If an adversary transmits a malicious data frame to an in-vehicle CAN in less than a 10-ms period, the network will generate a "CAN Bus off" error state indicating communication failure (This attack could be detected by an Intrusion Detection System). We also design a session key update protocol for security of the 32-bit MAC.

• Forward and Backward Key Secrecy: In the proposed security protocol, a session key is exposed by an external device that communicates with the in-vehicle CAN. However, it is difficult for the external device to acquire the keys of the forward session. If a connection between the external device and in-vehicle CAN expires, GECU broadcasts a Key Request Message including $\mathsf{Seed_{new}}$. Because the Key Request Message is encrypted by UK, the external device cannot know the keys to be used for the forward session. It is also difficult to acquire the key of a backward session. For example, although $\mathsf{EK_k}$, $\mathsf{AK_k}$, and $\mathsf{KEK_k}$ of the $\mathsf{k_{th}}$ session keys are exposed, it is impossible to acquire EK_{k-1}, AK_{k-1}, and KEK_{k-1} of the $\mathsf{k-1_{th}}$ session keys as there is no association between Seed_{k-1} and $\mathsf{Seed_k}$.

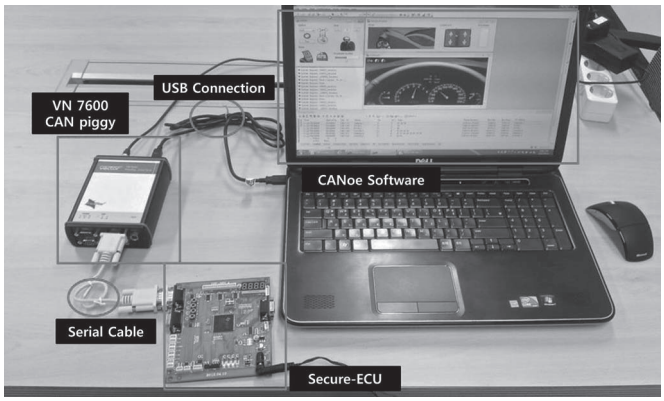

Fig. 10. Performance evaluation environment.

TABLE VI
TOOLS USED FOR THE SIMULATION

Product	Model Name	Note
Microcontroller	DSP F28335	60–150 MHz
Emulator	XDS100S	
Compiler	Code Composer	For Texas Instruments Embedded Processors
SW	CANoe	Network Simulation Tool for Vehicle
Connector	VAN 760 CAN Piggy	Interface Device

• KEY FRESHNESS: Keys used for every session are derived from a randomly generated value, hence, they have no association with each other. In other words, Seed_1, Seed_2, Seed_3, and Seed_n are different values.

• REPLAY ATTACK: In our proposed security protocol, a sender and a receiver manage the data frame counter. We used the data frame counter, which is synchronized and managed between the sender and receiver for generation of the MAC. As shown in Figs. 7 and 8, Section V, the sender uses the data frame counter in order to generate the MAC. As such, because the data frame counter is used for the generation of the MAC, our proposed security protocol is secure against a replay attack.

B. Performance Evaluation

For performance analysis of the proposed security protocol, we manufactured a Secure-ECU that has a similar functionality to that of a real ECU and then performed a hardware-based simulation. We also performed a simulation that interlocked the Secure-ECU with CANoe software. The simulation environments are as shown in Fig. 10. Table VI shows specifications of the equipment used for the evaluation.

1) Hardware-Based Evaluation (F28335 Microcontroller): We manufactured a Secure-ECU on a DSP-F28335 microcontroller of Texas Instruments for this evaluation. The execution time for encryption and authentication of a CAN data frame was measured by implementing the algorithms (AES-128, MAC) on the Secure-ECU firmware. Changing the CPU clock rates of the DSP-F28335 microcontroller to 150, 120, 90, and 60 MHz, we analyzed the resulting execution times of the proposed security protocol. For a more accurate evaluation, we repeated the protocol 1 000 000 times and obtained an average execution time, as shown in Fig. 11(a).

If the enhanced technique is used, the encryption and authentication of a CAN data frame can be performed within 378 μs when the CPU clock rate is 60 MHz. We note that if the proposed security protocol is implemented on Application Specific Integrated Circuits (ASICs), execution times will be faster than our implementation results [29], [30].

2) Software–Hardware-Based Evaluation: To construct an evaluation environment similar to that of a real in-vehicle CAN, we used CANoe of Vector Co. CANoe is the network simulation software used for developing or testing embedded systems for vehicles [31]. We constructed an evaluation environment using Secure-ECU, CANoe, and a VN7600 interface, as shown in Fig. 10. The proposed security protocol was also implemented on a CANoe virtual ECU node. We implemented our security protocol and built a DLL (Dynamic Linking Library) to apply it within CANoe.

However, we could not actually load a 32-bit MAC value to the Extended-ID and CRC fields because, as is generally the case for microcontrollers such as DSP-F28335, the unique functions of Extended-ID and CRC fields cannot be changed. Hence, we implemented the results of the hardware-based evaluation as an execution time delay for the software–hardware-based evaluation. After setting the execution time delay to happen before transmission and after reception of a data frame, we conducted the software–hardware-based evaluation.

• **Communication Response Time:** We measured the communication response time by connecting the Secure-ECU (receiver ECU) with CANoe. In this experiment, we varied the number of the virtual ECUs (sender ECUs) transmitting to the CAN data frame by 5, 10, 15, and 20 and measured communication response time of the Secure-ECU. Virtual ECUs broadcasted the CAN data frame with a cycle of 10 ms. After receiving the data frame, the Secure-ECU performed authentication and decryption, then transmitted a response CAN data frame. In Fig. 11(b), we plotted the response time of the Secure-ECU in terms of the number of virtual ECUs. When the CPU clock rate of the Secure-ECU is assumed to be 150 MHz, there is no significant difference between the general and modified CANs in implementing the proposed protocol. However, although not pictured in Fig. 11(b), when the clock rate was under 90 MHz, there was a loss in the received data frame when the number of virtual ECUs was more than 15. The loss of data frames occurred because the cycle of the received data frame was faster than the execution time needed for decryption and authentication in the data frame transmission and receipt processes.

However, we also conducted the performance evaluation experiment while setting the communication load much higher than that of a typical in-vehicle CAN. A typical in-vehicle CAN is divided into three subnetworks: 1) powertrain and chassis, 2) body electronics, and 3) infotainment. Each subnetwork is composed of less than 15 ECUs. In the case of a Volvo XC90, more than 40 ECUs are installed on more than two subnetworks. In particular, the largest subnetwork is the body electronics function, where 13 ECUs communicate with each other [5], [6]. Furthermore, in the newest ECUs used for vehicle development, microcontrollers with a computing power of more than 150 MHz have been installed [32] Hence, it is possible to

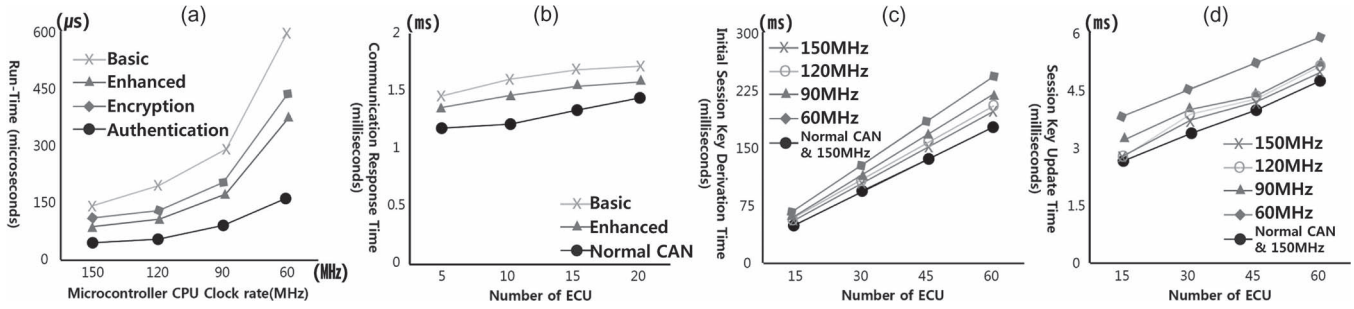

Fig. 11. (a) Execution time for encryption and authentication of a CAN data frame. (b) Communication response time of the in-vehicle CAN. (c) Initial session key derivation time. (d) Key update time.

use our proposed security protocol without data frame loss in the general in-vehicle CAN environment.

• **Initial session key derivation time:** Assuming the Secure-ECU was being used as GECU, we measured the initial session key derivation time in terms of the number of ECUs and the CPU clock rate. The CPU clock rate of GECU was fixed at 150 MHz. Fig. 11(c) shows the results of an average initial session key derivation time from our experiment. Our proposed security protocol used AKEP2 to derive the initial session key. In order to establish a session key in a secure manner, AKEP2 performed an authenticated three-way handshake. Once a certain ECU performed a three-way handshake with GECU, the communication response time delay occurred twice. In addition, as the next ECU began an initial session key derivation after confirming the last third of a three-way handshake, if N ECUs performed a three-way handshake with GECU, N-1 communication response time delays additionally occurred. In other words, the communication response time delay that happened when N ECUs performed an initial session key derivation with GECU was as follows.

$$(3N-1)*(\text{communication response time delay}). \quad (10)$$

The results of Fig. 11(c) analyzed in combination with those of Fig. 11(a), (b), show that the difference in initial session key derivation execution time arises from the difference of MAC function execution time. In the authenticated three-way handshake, the MAC function is used six times. When the ECU CPU clock rate is 150 or 60 MHz, the difference in MAC function execution time needed for the authenticated three-way handshake is about 720 μs. A comprehensive analysis of the results shows that the communication response time delay has a greater effect on the initial session key derivation time than the MAC function execution time. In addition, as shown in the results of Fig. 11(c), the time for 60 ECUs to complete the initial session key derivation process is less than 235 ms. Therefore, when applying our proposed security protocol to vehicles with low-performance ECUs, its availability may be sufficiently ensured.

• **Key Update Time:** We experimentally measured key update time in the same environment as that of the initial session keys derivation time. Fig. 11(d) shows the results, where it can be seen that key update can be performed within 6 ms. The key update time is similar regardless of CPU clock rate with the exception of 60 MHz because both the reception of a key request data frame and the generation of a key response data frame are performed in parallel on all ECUs. In other words, increases in key update time are proportional to the number of ECUs and communication speed rather than the ECUs' performance (CPU clock rate).

TABLE VII
COMPARATIVE EVALUATION CONDITIONS

CAN BUS Speed	1 Megabits per second	Fixed
MAC Function	32 bit MAC	Fixed
ECU Data Transmission Cycle	10 milliseconds	Fixed
Key Disclosure Delay	10, 7, 4, and 1 millisecond	Variable value
Number of ECU Nodes	20, 15, 10, and 5 nodes	Variable value

C. Security and Efficiency Comparison

Here, we compare our protocol to the protocol suggested in [14] [15]. For the convenience of description, our proposed security protocol is denoted as OURS and the protocol of [14] is denoted as EPSB (Efficient Protocol for Secure Broadcast) and the protocol of [15] is denoted as IDT&C(using ID Table & message Counter). There are two modes in EPSB: single master mode and multimaster mode. We only consider the single master mode since the multimaster mode requires a high bus load. In the single master mode of EPSB, one communication master conducts authentication of the data frames that every sender transmits. For a detailed comparison, we divided the single master mode of EPSB into EPSB − 1 and EPSB − 2. EPSB − 1 is a mode that contains both a message and MAC in one CAN data frame. EPSB − 2 is a mode that transmits two CAN data frames, one for a message and one for a MAC separately. IDT&C generates as many MAC messages as receivers in the communication group. Letting M be the number of ECUs in the communication group, the number of extra messages generated is $M*(M-1)$ in order for every ECU in the group to interchange messages.

We analyzed the required bus loads of OURS, EPSB-1, EPSB-2, and IDT&C using the evaluation environment given in Table VII. Key disclosure delays were applied only to EPSB-1 and EPSB-2. Fig. 12 shows the required bus loads in OURS, EPSB-1, EPSB-2, and IDT&C in terms of the number of ECU and the key disclosure delay. In a general CAN, the bus load has to remain under 50% of maximum to preserve a stable communication environment. As shown in Fig. 12(a), OURS keeps the bus load under 50%, although there are 20 ECUs because it neither requires an additional CAN data frame and nor uses key disclosure for message authentication.

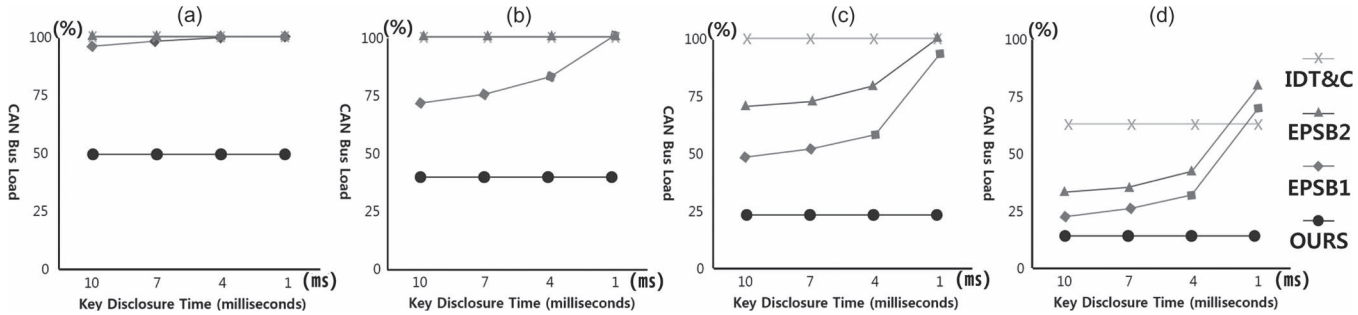

Fig. 12. Comparison between OURS, EPSB, and IDT&C at BUS Load. The number of ECUs that participated in communication were: (a) 20, (b) 15, (c) 10, and (d) 5.

TABLE VIII
SECURITY AND EFFICIENCY COMPARISON

	OURS	EPSB	IDT&C
Data Confidentiality	O	X	X
Data Authentication & Integrity	O	O	O
Connectivity with External Device	O	X	X
Increased BUS load	X	O	O
Resistance to Replay-attack	O	X	O

However, in the case of EPSB − 1 and EPSB − 2, bus loads increase by more than 50% due to the extra CAN data frames and key disclosures. EPSB can maintain a bus load under 50% only when there are less than five ECUs and the key disclosure delay is more than 4 ms. IDT&C also increases bus load rapidly since it additionally has to transmit as many MAC messages as receivers. IDT&C can maintain a bus load of 60% only when there were five ECUs in the communication group.

Table VIII shows the overall comparisons of security and efficiency of OURS, EPSB, and IDT&C. Detailed comparative evaluations of the data listed in Table VIII are as follows.

1) Security measures (data encryption and authentication).
2) Connectivity with external devices (i.e., key exchange with external devices).
3) Efficiency of the proposed technique (whether it may maintain a bus load of less than 50%).
4) Security from our attack-model. (Our attack-model uses the message replay attack. If security techniques can ensure security from the message replay attack, they are secured from our attack model.)

EPSB and IDT&C offer authentication for the CAN data frame but they do not consider the connection of external devices during vehicle operation. Furthermore, in EPSB and IDT&C, it is possible to analyze the meaning of certain data frames since confidentiality is not ensured. As shown in Fig. 12, EPSB and IDT&C increase bus load considerably. EPSB is also vulnerable to a message replay attack since a secret key shared between sender and communication master is not changed for every session. Thus, it is possible to perform a replay attack after eavesdropping on CAN data frames including MACs. IDT&C has disadvantages in that a MAC has to be transmitted additionally for message authentication and a message cannot be used before MAC verification. However, it is secure from replay attack. Based on the experimental findings, it is expected that security could be ensured with our proposed attack model when using IDT&C for communication groups of less than five ECUs. In contrast to those aforementioned, our proposed security protocol offers both security and efficiency.

OURS supports the connection of an external device. In addition, as aforementioned, OURS offers data frame confidentiality and authentication. OURS offers both confidentiality and authentication but rarely increases bus load, enabling real-time processing of CAN data frames. Furthermore, in OURS, a replay attack is impossible because a counter between the sender and receiver is managed and used for encryption and authentication of CAN data frames.

VIII. CONCLUSION

Recently, many studies on the vulnerability of in-vehicle CAN have been done. However, such attack models are unrealistic because they require significant effort and complex technology such as reverse engineering and carjacking. Thus, in this paper, we proposed an actual attack model using a malicious smartphone app in the connected car environment and demonstrated it through practical experiments. After demonstrating the attack model with an analysis of the vulnerability of in-vehicle CAN, we designed a security protocol that could be applied to the car environment. Furthermore, we analyzed the security and performance of the proposed security protocol through an evaluation based on both Secure-ECU and CANoe. In the future, we plan to improve the performance of the proposed security protocol with an implementation of the encryption and hash algorithms on hardware to optimize our security technology.

APPENDIX

The content of the news report at URL http://goo.gl/mo33ay is summarized as follows.

•(00:28)The application (app) is used for diagnosing a vehicle. •(00:31)By wirelessly connecting the electronic controller to the vehicle, the smartphone reveals information that is not displayed on the dashboard, such as vehicle breakdown information, accurate gas mileage information, and travel route information. •(00:43)There are over 200 vehicle diagnostic apps in the market that are used by many drivers. •(00:50)However, the problem is that if a hacker installs a malicious code to this App, he/she would be able to externally control the automobile. •(00:59)We carried out an experiment under the support by the research team, Korea University. •(01:04)As soon as the

attackers smartphone button is pressed, the dashboard immediately triggers the alarm. •(01:12)The ignition of the running car is suddenly turned off. •(01:19)The steering wheel goes randomly when using a smart parking system. •(01:26)More dangerous remote control is also possible. When the rapid acceleration button is pressed on the smartphone, the vehicles speed increases immediately, and it increases to 160 km/h in an instant with the vehicle lifted up. •(01:46)In this way, the hacker can control the automobile at will using the victims smartphone infected by his/her malicious codes.

REFERENCES

[1] A. Saad and U. Weinmann, "Automotive software engineering and concepts," *GI. Jahrestagung.*, vol. 34, pp. 318–319, 2003.

[2] E. Nickel, "IBM automotive software foundry," in *Proc. Conf. Comput. Sci. Autom. Ind.*, Frankfurt, Germany, 2003.

[3] M. Wolf, A. Weimerskirch, and T. Wollinger, "State of the art: Embedding security in vehicles," *EURASIP J. Embedded Syst.*, vol. 2007, no. 5, p. 1, 2007.

[4] R. Charette, This Car Runs on Code. [Online]. Available: http://www.spectrum.ieee.org/feb09/7649

[5] T. Nolte, H. Hansson, and L. L. Bello, "Automotive communications-past, current and future," in *Proc. IEEE Int. Conf. Emerging Technol. Factory Autom.*, 2005, vol. 1, pp. 992–999.

[6] K. H. Johansson, M. Torngren, and L. Nielsen, "Vehicle applications of controller area network," in *Handbook of Networked and Embedded Control Systems*. New York, NY, USA: Springer-Verlag, 2005, pp. 741–765.

[7] T. Hoppe and J. Dittman, "Sniffing/replay attacks on CAN buses: A simulated attack on the electric window lift classified using an adapted CERT taxonomy," in *Proc. Conf. Embedded Syst. Security*, 2007, pp. 1–6.

[8] T. Hoppe, S. Kiltz, and J. Dittmann, "Security threats to automotive CAN networks—Practical examples and selected short-term countermeasures," *Rel. Eng. Syst. Safety*, vol. 96, no. 1, pp. 11–25, Jan. 2011.

[9] K. Koscher *et al.*, "Experimental security analysis of a modern automobile," in *Proc. IEEE Security Privacy. Symp.*, Oakland, CA, USA, 2010, pp. 447–462.

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

[11] 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 *Proc. Conf. Veh. Technol.*, San Francisco, CA, USA, 2011, pp. 1–5.

[12] H. Schweppe *et al.*, "Securing Car2X applications with effective hardware software codesign for vehicular on-board networks," in *Proc. Conf. Autom. Security*, Berlin, Germany, 2011.

[13] D. K. Nilsson, U. E. Larson, and E. Jonsson, "Efficient in-vehicle delayed data authentication based on compound message authentication codes," in *Proc. Conf. IEEE 68th Int. Conf. Veh. Technol.*, Calgary, BC, Canada, 2008, pp. 1–5.

[14] B. Groza and S. Murvay, "Efficient protocols for secure broadcast in controller area networks," *IEEE Trans. Ind. Informa.*, vol. 9, no. 4, pp. 2034–2042, Nov. 2013.

[15] C. W. Lin and A. Sangiovanni Vincentelli, "Cyber-security for the Controller Area Network (CAN) communication protocol," in *Proc. Conf. IASE Int. Conf. Cyber Security*, 2012, pp. 344–350.

[16] P. Kleberger, T. Olovsson, and E. Jonsson, "Security aspects of the in-vehicle network in the connected car," in *Proc. IEEE Intell. Veh., Symp.*, 2011, pp. 528–533.

[17] BOSCH CAN, 2004, Webpage. [Online]. Available: www.can.bosch.com

[18] D. K. Nilsson, U. E. Larson, and E. Jonsson, "Creating a secure infrastructure for wireless diagnostics and software updates in vehicles," in *Proc. Conf. Comput. Safety, Rel., Security, Tyne, UK.*, Newcastle upon Tyne, U.K., 2008, pp. 207–220.

[19] S. Checkoway *et al.*, "Comprehensive experimental analyses of automotive attack surfaces," in *Proc. 19th Conf. USENIX Sec., Washington, DC*, 2011, p. 6.

[20] *IEEE Standard for Local and Metropolitan Area Networks Part 16 Air Interface for Fixed and Mobile Broadband Wireless Access Systems*, IEEE Std 802.16, 2009, IEEE Standard.

[21] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *Proc. Conf. CRYPTO*, 1993, pp. 232–249.

[22] J. M. Alfred, P. C. van O, and A. V. Scott, *Handbook of Applied Cryptography, Chapter-9-Hash Function*. Boca Raton, FL, USA: CRC Press, 1997, pp. 359–368, no. 4.

[23] S. You, M. Krage, and L. Jalics, "Overview of remote diagnosis and maintenance for automotive systems," in *Proc. SAE World Congr.*, Detroit, MI, USA, 2005, pp. 1–8.

[24] M. Shavit, A. Gryc, and R. Miucic, "Firmware Update Over The Air (FOTA) for automotive industry," in *Proc. Conf. Asia Pacific Autom. Eng.*, Hollywood, CA, USA, 2007.

[25] D. K. Nilsson and U. E. Larson, "Secure firmware updates over the air in intelligent vehicles," in *Proc. IEEE Int. Conf. Commun. Workshop*, Beijing, China, 2008, pp. 380–384.

[26] H. Hilpert, L. Thoroe, and M. Schumann, "Real-time data collection for product carbon footprints in transportation processes based on OBD2 and smartphones," in *Proc. Conf. Syst. Sci.*, 2011, pp. 1–10.

[27] J. Daemen and V. Rijmen, *The Design of Rijndael. AES-the Advanced Encryption Standard*. Berlin, Germany: Springer-Verlag, 2002.

[28] K. Yasuda, "Multilane HMAC: Security beyond the birthday limit," in *Proc. Conf. INDOCRYPT*, 2007, pp. 18–32.

[29] A. Hodjat and I. Verbauwhede, "Minimum area cost for a 30 to 70 Gbits/s AES processor," in *Proc. IEEE. Comput. Soc. Annu. Symp VLSI*, 2004, pp. 83–88.

[30] S. Mangard, M. Aigner, and S. Dominikus, "A highly regular and scalable AES hardware architecture," *IEEE Trans. Comput.*, vol. 52, no. 4, pp. 483–491, Apr. 2003.

[31] Vector, Webpage. [Online]. Available: www.vector-informatik.com

[32] Texas Instruments, Webpage. [Online]. Available: http://www.ti.com/TMS570

Samuel Woo received the M.S. degree in computer science from Dankook University, Seoul, Korea, in 2010. He is currently working toward the Ph.D. degree in information security in the Graduate School of Information Security, Korea University.

His research interests include cryptographic protocols in authentication, applied cryptography, security, and privacy in vehicular networks and controller area network security.

Hyo Jin Jo received the B.S. degree in industrial engineering from Korea University, Seoul, Korea, in 2009. He is currently working toward the Ph.D. degree in information security in the Graduate School of Information Security, Korea University.

His research interests include cryptographic protocols in authentication, applied cryptography, security, and privacy in ad hoc networks and smart car security.

Dong Hoon Lee (F'06) received the B.S. degree from the Department of Economics, Korea University, Seoul, Korea, in 1985, and the M.S. and Ph.D. degrees in computer science from University of Oklahoma, Norman, OK, USA, in 1988 and 1992, respectively.

Since 1993 he has been with the Faculty of Computer Science and Information Security, Korea University. Since 2004 he has been the President of the Ubiquitous Information Security Organization that has been supported by the BK21 Project in Korea. He is currently a Professor and the Vice Director of the Graduate School of Information Security with Korea University. His research interests include the design and analysis of cryptographic protocols in key agreement, encryption, signature, embedded device security, and privacy-enhancing technology.