Authentication for V2X Communication in an Open-Source Plug-and-Play Platform

Minhye Seo*
Duksung Women's University, Seoul, 01369, Republic of Korea
mhseo@duksung.ac.kr

Received: February 22, 2022; Accepted: April 14, 2022; Published: May 31, 2022

Abstract

Most cars today are made up of numerous electronic devices, which enable various user-friendly services such as autonomous driving. The car of the future will play a role as a platform for electronic systems, unlike fully built vehicles these days. In other words, on-vehicle devices with various functions produced by different companies will be freely attached to and detached from the vehicle, that is, in a plug-and-play (PnP) manner. However, if a PnP device produced by a malicious attacker is attached to a vehicle, the attacker can easily penetrate the in-vehicle network and take over the vehicle, which can pose a serious threat. In this paper, we propose for the first time PnP device authentication frameworks in the automotive open-source PnP platform, which allow only authorized PnP devices to access the in-vehicle network. We also develop a proof-of-concept (PoC) for the proposed authentication framework and show the experimental results of the PoC simulation to prove its feasibility.

Keywords: V2X communication, Entity authentication, Open-source Plug-and-Play, PnP platform

1 Introduction

In-car ICT technologies have become a new paradigm for developing cars of the future. Recently released vehicles are already equipped with hundreds of embedded computers, called electronic control units (ECUs), for various systems to provide user convenience and safety (e.g., advanced drived assistance system (ADAS), tire pressure monitoring system (TPMS), etc.), as well as various services to connect vehicles with customer electronic (CE) devices [1, 2]. That is, cars are evolving into electronics on wheels, similar to what happened to cellphones about 10 years ago. As a number of ICT technologies were deployed in cars, they have changed into subjects that collect and process various types of information generated while driving [3]. Cars have become more interconnected with each other, which allows them to discover themselves, connect and interact with surrounding cars (i.e., vehicle-to-vehicle (V2V) communication), as well as with the surrounding infrastructure (i.e., vehicle-to-infrastructure (V2I) communication) to build safe and convenient driving conditions. The recent advent of 5G technology has enabled cars to interact with everything around them (i.e., vehicle-to-everything (V2X) communication) in real time, which has made moves to new areas such as autonomous driving (or self-driving cars) [4].

Beyond automobile original equipment manufacturers (OEMs) producing fully-built vehicles, cars will be a kind of platform on which a number of commercial products are mounted. In other words, vehicles that have been developed in closed systems will be transformed into open-source platforms. In such a platform, various types of on-vehicle devices can be freely attached and detached from the vehicle in

Journal of Internet Services and Information Security (JISIS), volume: 12, number: 2 (May), pp. 1-20
DOI:*10.22667/JISIS.2022.05.31.001*

*Corresponding author: Department of of Cyber Security, Duksung Women's University, 33 Samyang-ro 144-gil, Dobong-gu, Seoul, 01369, Republic of Korea, Tel: +82-2-901-8840

a plug-and-play (PnP) method according to their purpose, and these PnP devices can be used universally regardless of specific type or manufacturer of the vehicle [5]. For example, PnP devices such as a variety of sensors and navigation can be selected according to the user's preference (e.g., specification, design, etc.), and these can be assembled to complete an autonomous vehicle. In this situation, whenever a PnP device is newly installed in a vehicle, authentication as to whether it is produced from a legitimate manufacturer must be preceded. Otherwise, a PnP device developed by an attacker for malicious purposes may be installed in the vehicle, causing serious problems such as malfunctioning or manipulating the vehicle to put the driver into a dangerous situation. A number of studies have already shown that if an attacker successfully breaks into the in-vehicle network, the attacker can control physical aspects of the vehicle such as turning the steering wheel to change direction, increasing/decreasing the speed abruptly, or stopping the brakes from working [6, 7, 8, 9, 10, 11].

Until recently, cars have been developed in closed structures and regarded as independent machines with no connectivity. In this sense, the controller area network (CAN) protocol, the most common protocol in automotive systems, was believed to be secure from external attacks, and consequently, was designed without consideration of security issues. However, as it has been shown that it is possible to remotely exploit vulnerabilities and physically control other vehicles (i.e., *hacking a car*), a number of studies have been actively conducted on authentication between ECUs in a vehicle [12, 13, 14, 15, 16, 17, 18]. The authentication here is about whether all devices in the vehicle are manufactured by the OEM and operate in a defined manner. However, as vehicles move the stage to an open-source platform, the scope of authentication needs to be much broader. PnP devices produced by various companies will be attached to the vehicle according to the driver's choice, and its proportion will increase more and more as the era of autonomous driving arrives. Therefore, in order to reflect the rapidly changing automobile industry structure, it is necessary to design authentication architectures for commercial PnP devices in an open-source platform and verify their validity. A variety of V2X services will come after authentication of all kinds of devices physically attached to the vehicle (producing/collecting necessary information for driving).

1.1 Contributions

In this paper, we propose new authentication architectures suitable for automotive open-source PnP environment for the first time. Several studies have tried to use PnP devices in a vehicle to provide additional functionalities [19, 20], but so far no research has considered the authentication of a PnP device in a vehicular environment. In Section 3, we designed four types of authentication frameworks in an open-source PnP platform, and each framework is different in terms of authentication technique and communication method (specifically how to establish a secure channel). In our authentication frameworks, we employ techniques based on symmetric-key encryption and asymmetric-key encryption for authentication, and a mobile communication network and SSL (Secure Socket Layer) communication for secure communication. Providing various options for PnP device authentication in the vehicular environment allows the proposed frameworks to be employed to a broader range of real-world applications.

We also draw operating guidelines for our authentication frameworks in automotive PnP platform in Section 4. We first analyze the characteristics of each of the four authentication frameworks and present the security requirements that each entity in the framework should provide based on the type of information it stores. This helps our frameworks to be securely deployed in real-world vehicular environments.

Finally, we develop a proof-of-concept (PoC) of the proposed authentication framework, *Sym-SSL*, with subsequent session key distribution protocol in Section 5, which can ensure the feasibility of our authentication architectures. We used CANoe simulator to build CAN-FD protocols for in-vehicle communications (between the PnP device and the gateway ECU). We then give experimental results of PoC

simulation. It can be inferred that the proposed authentication framework can be fully deployed in real-world automotive open-source PnP environment.

1.2 Organization

The remainder of this paper is organized as follows: In Section 2, we introduce an authentication architecture in automotive PnP Platform and basic assumptions required. We describe our proposed authentication mechanisms in automotive PnP platform in Section 3, followed by the operation guidelines and their comparisons in Section 4. Section 5 is dedicated to presenting the performance evaluation with implementation results.

2 System Model

2.1 Architecture

An authentication architecture in automotive PnP platform consists of four entities: certificate authority (managed by automotive OEMs), manufacturer of PnP devices, gateway ECU of a vehicle, and Plug-in PnP device. In an open-source PnP platform of a vehicle, any PnP device can be installed in the vehicle, and a gateway ECU of the vehicle should be able to verify whether the plugged-in PnP device is legitimate. To authenticate a legitimate PnP device, the vehicle's gateway ECU requests and receives information corresponding to the plugged-in PnP device from the PnP device manufacturer. Figure 1 shows the whole architecture of the automotive PnP platform for plug-in device authentication. In the manufacturing process of PnP devices, the manufacturer injects information (necessary for authentication) into the device. The certificate authority (CA) continuously manages the authentication system for mutual authentication and secure information transmission between the PnP device manufacturer and the vehicle's gateway ECU.

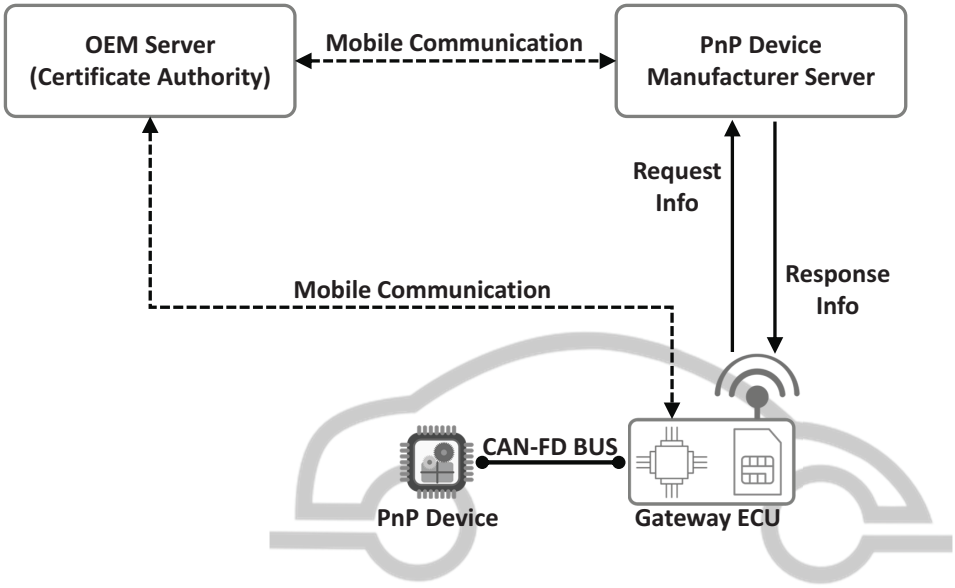

Figure 1: Authentication architecture in automotive PnP Platform

- **Certificate Authority**: A certificate authority (CA) is an authentication server managed by an automotive OEM (Original Equipment Manufacturer), and it creates and manages certificates for

both OEM-certified PnP device manufacturers and gateway ECUs in the vehicle. Certificates issued by CA are required for secure communication using SSL (Secure Socket Layer) between the PnP device manufacturer and the gateway ECU. If a mobile communication network (such as LTE, 5G) is used for communication, CA is not included in the system model.

- **PnP device manufacturer**: A PnP device manufacturer is a company that produces Plug-in devices and generates a (cryptographic) key required for authentication for each device. During the authentication process, the vehicle's gateway ECU requests the PnP device manufacturer for an authenticating value for the serial number (SN) of the plugged-in PnP device, and the manufacturer stores secret information to generate this value. The PnP device manufacturer receives a certificate (for SSL communication) from the CA and uses it to perform secure communication with the gateway ECU. The private key corresponding to the certificate should be stored in a secure manner.
- **Gateway ECU**: In the vehicle production process, an authenticating value of the vehicle is generated at the same time, and this value is stored in the gateway ECU of the vehicle. The stored value varies according to the communication method with the PnP device manufacturer, and the existence of HSM (Hardware Security Module) is determined according to this value. If communication between the PnP device manufacturer and the gateway ECU is performed over a mobile communication network, the value (stored in the gateway ECU) becomes a white list for the PnP device manufacturer managed by the OEM. If the communication is performed using SSL, the stored value becomes a certificate (for SSL communication) issued by the CA. The private key corresponding to the certificate should be stored securely in the HSM in the gateway ECU.
- **PnP device**: PnP device manufacturers inject authenticating values into the device in the production process, store the values in a secure manner using HSM if necessary. The stored authenticating value is associated with the serial number (SN) of the PnP device, and upon request of the gateway ECU, the PnP device manufacturer restores the authenticating value (in the device) using the serial number of the PnP device.

2.2 Assumptions

Basic assumptions are required for entity authentication in the automotive PnP platform.

- In-vehicle communication: In-vehicle communication (such as CAN and CAN-FD) is robust against man-in-the-middle (MITM) attack and denial-of-service (DoS) attack.
- Secure storage: PnP devices and gateway ECUs have a secure storage, HSM, which is robust against various attacks with physical access and attacks on main memory or discs (e.g. memory dump).
- Authentication: The gateway ECU authenticates PnP devices newly installed (or re-installed) in the vehicle. Once authentication is completed, there is no attempt to re-authenticate until the PnP device is removed from the vehicle, that is, no authentication attempt occurs every time the engine is turned on. We assume that the vehicle has a physical security device that can check whether the PnP device has ever been removed even when the engine is turned off. Therefore, even if the PnP device is removed and then re-installed while the engine is turned off, the gateway ECU can grasp the situation and re-authenticate the plugged-in PnP device. Also, queries (i.e., challenge messages) generated by the gateway ECU in the authentication process are unpredictable, which can guarantee the freshness of the authenticating value and thereby prevent a replay attack. Finally, we assume that a secure communication channel (using mobile communication network or SSL communication) is established between the PnP device manufacturer and the gateway ECU.

3 Authentication in Automotive PnP Platform

In this section, we propose four authentication frameworks for plug-in devices in automotive PnP platform. Each authentication framework uses a different method for secure communication and authentication between PnP devices and gateway ECUs. For secure communication between the PnP device manufacturer and the gateway ECU, a mobile communication network and SSL (Secure Socket Layer) communication can be used. For authentication between the PnP device and the gateway ECU, techniques based on symmetric-key encryption and asymmetric-key encryption can be used. Table 1 shows the four framework categories according to the two criteria above.

		(Secure communication)	
		Mobile network	**SSL**
(Authentication technique)	**Symmetric**	Sym-Mob (3.1)	Sym-SSL (3.2)
	Asymmetric	Asy-Mob (3.3)	Asy-SSL (3.4)

Table 1: Authentication frameworks in automotive PnP platform

3.1 Sym-Mob framework

In the Sym-Mob framework, the vehicle's gateway ECU and the plugged-in PnP device have the same secret key, and the gateway ECU uses this symmetric key to authenticate the PnP device. For authentication, the Sym-Mob framework deploys ISO/IEC 9798-2, a standard protocol for one-way authentication [21]. The gateway ECU requests the PnP device manufacturer to receive the secret key corresponding to the plugged-in PnP device. At this time, a mobile communication network is used for secure communication between the PnP device manufacturer and the gateway ECU. Figure 2 illustrates the procedure of Sym-Mob framework. The Sym-Mob framework consists of three phases: *Setup*, *Preparation*, and *Authentication*.

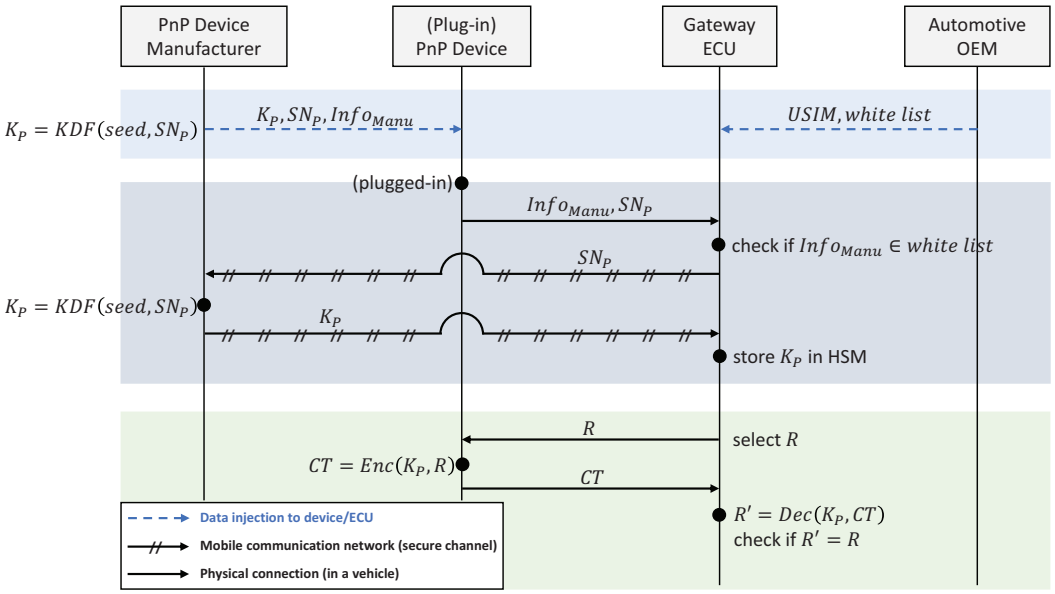

Figure 2: The procedure of Sym-Mob framework

1. **Setup phase**: In the setup phase, an environment for authentication is established. When manufacturing a PnP device, the manufacturer generates a secret key K_P for each device and stores it in the HSM (Hardware Security Module) within the PnP device. The secret key is generated using the Key Derivation Function (KDF), which takes the secret information, *seed*, of the manufacturer and the device's serial number, SN_P, as inputs and outputs the secret key K_P. The PnP device manufacturer stores the *seed* in a secure manner, and deletes the device's serial number SN_P and secret key K_P. Automotive OEMs (Original Equipment Manufacturers) insert USIMs (Universal Subscriber Identity Modules) in each gateway ECU during vehicle production to use mobile communication networks. In addition, the white list of PnP device manufacturers certified by the OEM is also stored in the gateway ECU.

2. **Preparation phase**: When the PnP device is installed (or plugged-in) on the vehicle, the preparation phase begins. In the preparation phase, the gateway ECU receives the secret key K_P of the PnP device from the PnP device manufacturer through the following steps:

   (a) When a PnP device is installed on the vehicle, it transmits the information of the PnP device manufacturer, $Info_{Manu}$, and its serial number SN_P to the gateway ECU.

   (b) The gateway ECU check whether the transmitted information $Info_{Manu}$ is included in the white list, which is a list of manufacturers certified by the OEM and stored in the Setup phase. If $Info_{Manu}$ is included in the white list, the gateway ECU uses $Info_{Manu}$ to connect communication with the PnP device manufacturer (via mobile communication network).

   (c) The gateway ECU transmits the serial number SN_P of the plugged-in device to the PnP device manufacturer.

   (d) The PnP device manufacturer generates a secret key K_P using the received serial number SN_P and the stored secret information *seed*, and transmits K_P to the gateway ECU in a secure manner.

   (e) The gateway ECU stores the secret key K_P in the HSM for subsequent authentication.

   Note that the PnP device manufacturer and the gateway ECU communicate using a mobile communication network and authenticate each other through a mobile carrier.

3. **Authentication phase**: In the authentication phase, one-way authentication is performed. The gateway ECU authenticates the PnP device using information stored on both sides (such as secret keys) through the following steps:

   (a) The gateway ECU selects a random number R as a challenge message and transmits it to the PnP device.

   (b) The PnP device encrypts the challenge R with its secret key K_P to generate a ciphertext $CT = Enc(K_P, R)$, and transmits CT to the gateway ECU.

   (c) The gateway ECU decrypts the ciphertext CT with the secret key K_P transmitted from the PnP device manufacturer: $Dec(K_P, CT) = R'$. If $R' = R$, the gateway ECU authenticates the plugged-in PnP device as a legitimate one.

3.2 Sym-SSL framework

The Sym-SSL framework is similar to the Sym-Mob framework (in Section 3.1), except that SSL communication is used to establish a secure channel between the vehicle's gateway ECU and the PnP device

manufacturer. In the Sym-SSL framework, the gateway ECU and the PnP device manufacturer authenticate each other based on trust in the certificate authority CA. Before establishing a secure channel, they either download a CRL (Certificate Revocation List) from the CA or perform the OCSP (Online Certificate Status Protocol) to verify that the other is a legitimate entity. The Sym-SSL framework consists of three phases: *Setup*, *Preparation*, and *Authentication*.

1. **Setup phase**: In the setup phase, an environment for authentication is established. When manufacturing a PnP device, the manufacturer generates a secret key K_P for each device and stores it in the HSM within the PnP device. The secret key is generated using the Key Derivation Function (KDF), which takes the secret information *seed* of the manufacturer and the device's serial number SN_P as inputs and outputs the secret key K_P. The PnP device manufacturer stores the *seed* in a secure manner, and deletes the device's serial number SN_P and secret key K_P. The automotive OEM receives a certificate of each gateway ECU $Cert_{GW}$ (for SSL communication) from the CA and inserts it into the corresponding gateway ECU. And the PnP device manufacturer also receives a certificate $Cert_{Manu}$ (for SSL communication).

2. **Preparation phase**: When the PnP device is installed (or plugged-in) on the vehicle, the preparation phase begins. In the preparation phase, the gateway ECU receives the secret key K_P of the PnP device from the PnP device manufacturer through the following steps:

   (a) When a PnP device is installed on the vehicle, it transmits the information of the PnP device manufacturer, $Info_{Manu}$, and its serial number SN_P to the gateway ECU.

   (b) The gateway ECU and the PnP device manufacturer verify the validity of each other's certificates by using the CRL (downloaded from the CA) or by performing the OCSP with the CA. If both certificates are valid, they establish a secure channel using SSL communication.

   (c) The gateway ECU transmits the serial number SN_P of the plugged-in device to the PnP device manufacturer.

   (d) The PnP device manufacturer generates a secret key K_P using the received serial number SN_P and the stored secret information *seed*, and transmits K_P to the gateway ECU in a secure manner.

   (e) The gateway ECU stores the secret key K_P in the HSM for subsequent authentication.

3. **Authentication phase**: In the authentication phase, one-way authentication is performed in the same way as in the Sym-Mob framework. For details, see the authentication phase in Section 3.1.

3.3 Asy-Mob framework

In the Asy-Mob framework, the PnP device has a public/private key pair, and the gateway ECU of a vehicle authenticates the plugged-in PnP device using the public key of the PnP device. For authentication, the Asy-Mob framework deploys ISO/IEC 9798-3, a standard protocol for one-way authentication [22]. The gateway ECU requests the PnP device manufacturer to receive the public key corresponding to the plugged-in PnP device. At this time, a mobile communication network is used for secure communication between the PnP device manufacturer and the gateway ECU. The Asy-Mob framework consists of three phases: *Setup*, *Preparation*, and *Authentication*.

1. **Setup phase**: In the setup phase, an environment for authentication is established. When manufacturing a PnP device, the manufacturer generates a public/private key pair (PK_P, SK_P) for each device and injects the key pair into the device. The private key SK_P is stored in the HSM within

the PnP device. In addition, the manufacturer stores a serial number SN_P and the public key PK_P of the device in the database. In the vehicle production process, the OEM inserts both a USIM to use mobile communication network and a white list of PnP device manufacturers certified by the OEM into the gateway ECU.

2. **Preparation phase**: When the PnP device is installed (or plugged-in) on the vehicle, the preparation phase begins. In the preparation phase, the gateway ECU receives the public key PK_P of the PnP device from the PnP device manufacturer through the following steps:

   (a) When a PnP device is installed on the vehicle, it transmits the information of the PnP device manufacturer, $Info_{Manu}$, and its serial number SN_P to the gateway ECU.

   (b) The gateway ECU check whether the transmitted information $Info_{Manu}$ is included in the white list, which is a list of manufacturers certified by the OEM and stored in the Setup phase. If $Info_{Manu}$ is included in the white list, the gateway ECU uses $Info_{Manu}$ to connect communication with the PnP device manufacturer (via mobile communication network).

   (c) The gateway ECU transmits the serial number SN_P of the plugged-in device to the PnP device manufacturer.

   (d) The PnP device manufacturer checks whether there is the transmitted serial number SN_P is included in the database. If so, the manufacturer transmits the public key PK_P corresponding to SN_P to the gateway ECU.

   (e) The gateway ECU stores the public key PK_P for subsequent authentication.

   Note that the PnP device manufacturer and the gateway ECU communicate using a mobile communication network and authenticate each other through a mobile carrier.

3. **Authentication phase**: In the authentication phase, one-way authentication is performed. The gateway ECU authenticates the PnP device using information stored on both sides through the following steps:

   (a) The gateway ECU selects a random number R_1 as a challenge message and transmits it to the PnP device.

   (b) The PnP device selects a random number R_2, and generates a signature $\sigma = Sign\big(SK_P, (R_1, R_2, ID_{GW})\big)$ by signing random numbers R_1 and R_2, and the ID of the gateway ECU ID_{GW} with its private key SK_P. Then the PnP device transmits $(R_1, R_2, ID_{GW}, \sigma)$ to the gateway ECU.

   (c) The gateway ECU verifies the signature σ with the public key PK_P of the PnP device and (R_1, R_2, ID_{GW}): $Verify\big(PK_P, (R_1, R_2, ID_{GW}), \sigma\big) = 1$ *or* 0. If the verification succeeds, the gateway ECU authenticates the plugged-in PnP device as a legitimate one.

3.4 Asy-SSL framework

The Asy-SSL framework is similar to the Asy-Mob framework (in Section 3.3), except that SSL communication is used to establish a secure channel between the vehicle's gateway ECU and the PnP device manufacturer. In the Asy-SSL framework, the gateway ECU and the PnP device manufacturer authenticate each other based on trust in the certificate authority CA. Before establishing a secure channel, they either download a CRL (Certificate Revocation List) from the CA or perform the OCSP (Online Certificate Status Protocol) to verify that the other is a legitimate entity. The Asy-SSL framework consists of three phases: *Setup*, *Preparation*, and *Authentication*.

1. **Setup phase**: In the setup phase, an environment for authentication is established. When manufacturing a PnP device, the manufacturer generates a public/private key pair (PK_P, SK_P) for each device and injects the key pair into the device. The private key SK_P is stored in the HSM within the PnP device. In addition, the manufacturer stores a serial number SN_P and the public key PK_P of the device in the database. The automotive OEM receives a certificate of each gateway ECU $Cert_{GW}$ (for SSL communication) from the CA and inserts it into the corresponding gateway ECU. And the PnP device manufacturer also receives a certificate $Cert_{Manu}$ (for SSL communication).

2. **Preparation phase**: When the PnP device is installed (or plugged-in) on the vehicle, the preparation phase begins. In the preparation phase, the gateway ECU receives the public key PK_P of the PnP device from the PnP device manufacturer through the following steps:

   (a) When a PnP device is installed on the vehicle, it transmits the information of the PnP device manufacturer, $Info_{Manu}$, and its serial number SN_P to the gateway ECU.

   (b) The gateway ECU and the PnP device manufacturer verify the validity of each other's certificates by using the CRL (downloaded from the CA) or by performing the OCSP with the CA. If both certificates are valid, they establish a secure channel using SSL communication.

   (c) The gateway ECU transmits the serial number SN_P of the plugged-in device to the PnP device manufacturer.

   (d) The PnP device manufacturer checks whether there is the transmitted serial number SN_P is included in the database. If so, the manufacturer transmits the public key PK_P corresponding to SN_P to the gateway ECU.

   (e) The gateway ECU stores the public key PK_P for subsequent authentication.

3. **Authentication phase**: In the authentication phase, one-way authentication is performed in the same way as in the Asy-Mob framework. For details, see the authentication phase in Section 3.3.

4 Operation Guidelines and Comparisons

In this section, we present operation guidelines for each of four authentication frameworks in automotive PnP platform (proposed in Section 3). Table 2 shows all the information stored by each entity in authentication frameworks and their security requirements in terms of confidentiality, integrity, and availability. If confidentiality is to be provided, the information must be transmitted over a secure channel and stored in a secure storage (such as HSM). Note that all information used for authentication must provide integrity and availability.

In addition, we analyze their characteristics through comparisons. Authentication frameworks in automotive PnP platform are distinguished by two criteria: *authentication technique* and *secure communication method*.

- The authentication technique refers to a method by which the gateway ECU authenticates the plugged-in PnP device, and authentication protocols based on symmetric key cryptography and asymmetric (or public) key cryptography can be used.

  In the symmetric key-based authentication framework (i.e., *Sym-Mob* and *Sym-SSL*), a PnP device manufacturer generates the (symmetric) secret key when producing a PnP device, and injects it into the HSM (Hardware Security Module), a secure storage in the device. When a PnP device is installed (or plugged-in) on a vehicle, the device transmits the secret key (stored in the HSM) to

Sym-Mob framework					
Entity	**Stored information**		**Confidentiality**	**Integrity**	**Availability**
PnP Device Manufacturer	$seed$	- Secret information to generate K_P	O	O	O
Gateway ECU	white list	- List of PnP device manufacturers certified by OEM	X	O	O
	K_P	- Secret key to authenticate PnP device	O	O	O
PnP Device	$Info_{Manu}$	- Information of PnP device manufacturer (e.g., name of company, IP address)	X	O	O
	SN_P	- Serial number of PnP device	X	O	O
	K_P	- Secret key to authenticate PnP device	O	O	O
Sym-SSL framework					
Entity	**Stored information**		**Confidentiality**	**Integrity**	**Availability**
Certificate Authority (CA)	CRL	- Certificate Revocation List issued by CA	X	O	O
PnP Device Manufacturer	$seed$	- Secret information to generate K_P	O	O	O
	$Cert_{Manu}$	- Certificate of PnP Device Manufacturer (for SSL communication with Gateway ECU)	O	O	O
Gateway ECU	$Cert_{GW}$	- Certificate of Gateway ECU (for SSL communication with PnP Device Manufacturer)	O	O	O
	K_P	- Secret key to authenticate PnP device	O	O	O
PnP Device	$Info_{Manu}$	- Information of PnP device manufacturer (e.g., name of company, IP address)	X	O	O
	SN_P	- Serial number of PnP device	X	O	O
	K_P	- Secret key to authenticate PnP device	O	O	O
Asy-Mob framework					
Entity	**Stored information**		**Confidentiality**	**Integrity**	**Availability**
PnP Device Manufacturer	(SN_P, PK_P)	- Database of pairs of serial numbers and public keys of PnP devices	X	O	O
Gateway ECU	white list	- List of PnP device manufacturers certified by OEM	X	O	O
	PK_P	- Public key of (plugged-in) PnP device	X	O	O
PnP Device	$Info_{Manu}$	- Information of PnP device manufacturer (e.g., name of company, IP address)	X	O	O
	SN_P	- Serial number of PnP device	X	O	O
	PK_P	- Public key of PnP device	X	O	O
	SK_P	- Private key of PnP device	O	O	O
Asy-SSL framework					
Entity	**Stored information**		**Confidentiality**	**Integrity**	**Availability**
Certificate Authority (CA)	CRL	- Certificate Revocation List issued by CA	X	O	O
PnP Device Manufacturer	(SN_P, PK_P)	- Database of pairs of serial numbers and public keys of PnP devices	X	O	O
	$Cert_{Manu}$	- Certificate of PnP Device Manufacturer (for SSL communication with Gateway ECU)	O	O	O
Gateway ECU	$Cert_{GW}$	- Certificate of Gateway ECU (for SSL communication with PnP Device Manufacturer)	O	O	O
	PK_P	- Public key of (plugged-in) PnP device	X	O	O
PnP Device	$Info_{Manu}$	- Information of PnP device manufacturer (e.g., name of company, IP address)	X	O	O
	SN_P	- Serial number of PnP device	X	O	O
	PK_P	- Public key of PnP device	X	O	O
	SK_P	- Private key of PnP device	O	O	O

Table 2: Operation guidelines for authentication in automotive PnP platform

the gateway ECU, and then the gateway ECU performs authentication with the PnP device manufacturer using this secret key. Therefore, the PnP device manufacturer should store the secret keys of all PnP devices it produces in a secure manner, which demands significant storage overhead. However, by using a key derivation function (KDF), the secret key of each device can be extracted using a small-size secret information *seed* and serial number SN_P of each PnP device. Since a *seed* alone can generate secret keys injected into all PnP devices, it must be stored securely. In terms of communication overhead, the symmetric key-based authentication technique is more efficient in that the size of transmitted data between the gateway ECU and PnP device (for authentication) is much smaller.

In the asymmetric key-based authentication framework (i.e., *Asy-Mob* and *Asy-SSL*), a PnP device manufacturer generates a pair of public/private keys when producing a PnP device, and injects the private key into the HSM in the device. When performing authentication, a PnP device manufacturer has to transmit the public key of plugged-in PnP device to the gateway ECU. Therefore, the

PnP device manufacturer should store the public keys of all PnP devices it produces. PnP device manufacturers should store serial numbers and public keys of all PnP devices they produce in a database, which incurs significant storage overhead. However, the information in the database does not need to be stored in a secure manner. In other words, even if the information stored in the database is exposed, no security issues arise.

Table 3 presents the difference between symmetric key-based and asymmetric key-based authentication frameworks in automotive PnP platform.

		Symmetric key based	**Asymmetric key based**
PnP Device Manufacturer	**Stored information**	*seed*	Database of (SN_P, PK_P)
	Confidentiality	O	X
Gateway ECU - PnP Device	**Communication overhead**	low (efficient)	high (inefficient)

Table 3: Comparison between symmetric key-based and asymmetric key-based authentication frameworks

- The secure communication method refers to a method of establishing a secure channel between the gateway ECU and the PnP device manufacturer, and a mobile communication network and SSL communication can be used.

  In the authentication framework using the mobile communication network (i.e., *Sym-Mob* and *Asy-Mob*), the PnP device manufacturer and the gateway ECU authenticate each other through a mobile carrier. The gateway ECU contacts the PnP device manufacturer with the white list inserted in the vehicle via mobile communication network. In order to use the mobile communication network, the OEM inserts the USIM issued by the mobile carrier into the gateway ECU when manufacturing vehicles.

  In the authentication framework using the SSL communication (i.e., *Sym-SSL* and *Asy-SSL*), the PnP device manufacturer and the gateway ECU authenticate each other with certificates issued by a certificate authority (CA). CA (managed by OEM) issues certificates for SSL communication, and PnP device manufacturers and gateway ECUs download CRLs (Certificate Revocation Lists) or perform OCSP (Online Certificate Status Protocol) before establishing a secure channel to verify the validity of the certificate of the other. PnP device manufacturers should be issued a certificate before the PnP devices they produce are released into the market, and OEMs should be issued a certificate when manufacturing vehicles so that it can be inserted into the vehicle's gateway ECU.

5 Performance Evaluation

In this section, we provide a Proof of Concept (PoC) in laboratory settings to demonstrate the feasibility of the proposed authentication framework. We simulate the *Sym-SSL* framework (proposed in Section 3.2) and present the implementation results.

5.1 Simulation configuration

The simulation configuration diagram of the Sym-SSL framework is shown in Figure 3, and consists of a total of four entities: certificate authority (OEM), PnP device manufacturer, gateway ECU, and PnP device. Each entity plays the following role in the simulation.

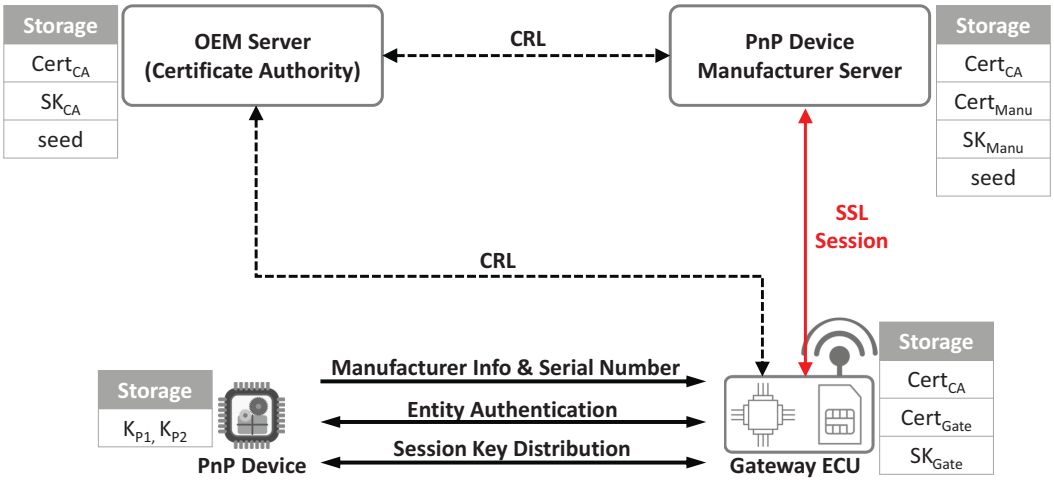

Figure 3: Simulation configuration of Sym-SSL framework

- **Certificate authority (OEM)**: The certificate authority (CA) managed by OEM is implemented in C as a window-based application program. The CA maintains a certificate revocation list (CRL) and signs the CRL with its own private key. When the PnP device manufacturer and the gateway ECU send a request on CRL, the CA sends the (latest version of) CRL and its signature, respectively.
- **PnP device manufacturer**: The PnP device manufacturer is implemented in C as a window-based application program. The PnP device manufacturer exchanges certificates with the gateway ECU of a vehicle, and verifies the validity of the gateway ECU's certificate using the CRL from CA. If the verification succeeds, the PnP device manufacturer first establishes a secure channel with the gateway ECU via SSL communication, generates secret information, and transmits it to the gateway ECU.
- **Gateway ECU**: The gateway ECU is implemented in C as a window-based application program. The gateway ECU exchanges certificates with the PnP device manufacturer, and verifies the validity of the PnP device manufacturer's certificate using the CRL from CA. If the verification succeeds, the gateway ECU first establishes a secure channel with the PnP device manufacturer via SSL communication and stores the secret information transmitted from the PnP device manufacturer.

  In addition, we emulate the gateway ECU as a virtual ECU simulated by CANoe for PnP device authentication and session key distribution. The gateway ECU first authenticates the PnP device using the standard protocol for symmetric key-based authentication (ISO/IEC 9798-2) [21]. And, if the authentication succeeds, the gateway ECU generates a session key and transmits it to the PnP device using the standard protocol for session key distribution (ISO/IEC 11770-2) [23].
- **PnP device**: The PnP device is emulated as a virtual ECU on the CANoe simulator. The PnP device is first authenticated by the gateway ECU using the standard protocol for symmetric key-

based authentication (ISO/IEC 9798-2) [21]. And, if the authentication succeeds, the PnP device receives the session key (generated by the gateway ECU) from the gateway ECU using the standard protocol for session key distribution (ISO/IEC 11770-2) [23], and stores it in a secure manner.

5.2 Notations and Consideration

The overall notations used in PoC are shown in Table 4. And the PoC was implemented based on the following considerations.

Notation	Description
$Cert_X$	A certificate of entity X ($Cert_X$ stores X's public key)
SK_X	A private key of entity X (SK_X is stored in secure storage such as HSM)
CRL	A certificate revocation list issued and managed by CA
$Info_{Manu}$	A unique information of PnP device manufacturer (e.g., name of company, IP address, URL)
SN	A (unique) serial number of PnP device
$Seed$	A secret information of PnP device manufacturer (Input to KDF for generating K_{P1} and K_{P2})
K_{P1}	A long-term secret key for PnP device authentication
K_{P2}	A long-term secret key for distribution of session keys
SSK	A session key generated by Gateway ECU
KDF	A key derivation function for generating K_{P1} and K_{P2}

Table 4: Notations used in PoC

- **Gateway ECU implementation**: In the proposed authentication framework, the gateway ECU can communicate with both inside and outside the vehicle. However, in the PoC simulation, the gateway ECU was implemented as two independent parts: a module for in-vehicle communication and a module for outside communication. The module for in-vehicle communication can use CAN-FD protocols, and communicate with the PnP device on the CANoe simulator to authenticate the PnP device and distribute a session key. The module for outside communication can communicate with the CA and the PnP device manufacturer, and perform the preparation phase for authentication. It is assumed that the gateway ECU has the capability to process public key-based operations in real time.
- **Absence of HSM**: In the proposed authentication framework, the HSM is used in the gateway ECU to securely store the secret key (for authentication) and the session key (for secure communication). However, HSM cannot be implemented in the experiment since the in-vehicle network communication is implemented on CANoe simulation. Therefore, secret information (such as K_{P1}, K_{P2}, SSK) to be stored in HSM is stored in general storage.
- **Certificate issuance**: In the experiment, we assume that all entities in the authentication framework have been issued certificates from the CA through due process. Therefore, we regard each entity's private key generation and certificate issuance as preliminary work before PoC simulation. In practice, the OpenSSL library was used to implement private key generation, certificate issuance from the CA, and CRL update by the CA, and the RSA-1024 algorithm was used to generate public keys of certificates.

- **Transmission of unique information**: When the PoC simulation starts, the PnP device transmits unique information (i.e., $Info_{Manu}$ and *SN*) to the gateway ECU using the module for in-vehicle communication. And then, the gateway ECU refers to $Info_{Manu}$ and tries to communicate with the PnP device manufacturer using the module for outside communication, and transmits *SN* after establishing a secure channel via SSL communication. However, in the experiment, since the two communication modules of the gateway ECU are separated and developed in different environments, $Info_{Manu}$ and *SN* are fixed to specific IP address and 8-byte data, respectively. In other words, transmission of unique information is excluded in the preparation phase of PoC simulation.

Phase	Description
Preparation phase	- Certificate exchange - CRL request/verification & Certificate validation (PnP device manufacturer) - CRL request/verification & Certificate validation (Gateway ECU) - SSL communication & Secret key distribution
Authentication phase	- PnP device authentication (ISO/IEC 9798-2)
Session key distribution phase	- Session key generation and distribution (ISO/IEC 11770-2)

Table 5: Procedure of PoC simulation

5.3 Proof of Concept (PoC)

The PoC simulation is divided into three phases: the preparation phase, the authentication phase, and the session key distribution phase. In addition to the preparation and authentication phases in the Sym-SSL framework, the session key distribution phase (between the gateway ECU and the PnP device) has been implemented using the standard protocol ISO/IEC 11770-2 [23]. The procedure of PoC simulation is shown in Table 5.

5.3.1 Preparation phase

The preparation phase is performed by three entities: CA (OEM), PnP device manufacturer, and gateway ECU. Each entity is implemented as a window-based application program, and carries out TCP/IP and SSL communications. Note that we assume that all entities have been issued certificates from the CA in advance, *CRL* is updated and managed by the CA, and the PnP device manufacturer securely stores *seed* for secret key generation, as described in Section 5.2. The preparation phase can be divided into 4 sub-processes as follows.

- **Certificate exchange**

  1. The gateway ECU checks $Info_{Manu}$, and transmits *SN* and $Cert_{GW}$ to the PnP device manufacturer;
  2. The PnP device manufacturer stores *SN* and $Cert_{GW}$;
  3. The PnP device manufacturer transmits $Cert_{Manu}$ to the gateway ECU;

4. The gateway ECU stores $Cert_{Manu}$.

- **CRL request/verification & Certificate validation (PnP device manufacturer)**

1. The PnP device manufacturer transmits a CRL request to the CA;
2. CA generates a signature $\sigma = Sign(SK_{CA}, CRL)$ (signed with its private key SK_{CA}) of a latest version of *CRL*, and transmits (CRL, σ) to the PnP device manufacturer;
3. The PnP device manufacturer verifies σ using $Cert_{CA}$ and *CRL*, and stores *CRL* if the verification succeeds;
4. The PnP device manufacturer verifies $Cert_{GW}$ with *CRL*. For certificate validation, x509_verify_cert() function in OpenSSL was used.

- **CRL request/verification & Certificate validation (Gateway ECU)**

1. The gateway ECU transmits a CRL request to the CA;
2. CA generates a signature $\sigma = Sign(SK_{CA}, CRL)$ (signed with its private key SK_{CA}) of a latest version of *CRL*, and transmits (CRL, σ) to the PnP device manufacturer;
3. The gateway ECU verifies σ using $Cert_{CA}$ and *CRL*, and stores *CRL* if the verification succeeds;
4. The gateway ECU verifies $Cert_{Manu}$ with *CRL*. For certificate validation, x509_verify_cert() function in OpenSSL was used.

- **SSL communication & Secret key distribution**

1. The PnP device manufacturer and the gateway ECU use *CertManu* and $Cert_{GW}$ to establish a secure channel via SSL communication, which is implemented using the OpenSSL library. After that, data can be transmitted securely without a separate encryption process (using SSL_write and SSL_read);
2. The PnP device manufacturer computes $KDF(seed, SN)$ to generate (K_{P1}, K_{P2}). The PKCS5_PBKDF2_HMAC_SHA1 in OpenSSL was used as the KDF, and the output length was set to 256-bit;
3. The PnP device manufacturer transmits K_{P1} and K_{P2} to the gateway ECU;
4. The gateway ECU stores K_{P1} and K_{P2} in a secure manner.

5.3.2 Authentication phase

The authentication phase is performed by the gateway ECU and the PnP device. Each entity is emulated on the CANoe simulator, and the gateway ECU authenticates the PnP device using the standard protocol ISO/IEC 9798-2 [21]. Note that both entities store the same secret keys (K_{P1}, K_{P2}).

- **PnP device authentication**

1. The gateway ECU selects a random number R_1 of length 128-bit and transmits R_1 to the PnP device;
2. The PnP device generates a ciphertext $CT_1 = Enc(K_{P1}, R_1)$ using AES encryption, and transmits CT_1 to the gateway ECU;
3. The gateway ECU computes $R'_1 = Dec(K_{P1}, CT_1)$ using AES decryption, and check if $R'_1 = R_1$. If $R'_1 = R_1$, the gateway ECU authenticates the PnP device as a legitimate one. (Authentication completed)

5.3.3 Session key distribution phase

The session key distribution phase is performed by the gateway ECU and the PnP device. Each entity is emulated on the CANoe simulator, and the session key (between the gateway ECU and the PnP device) is established using the standard protocol ISO/IEC 11770-2 [23].

- **Session key generation and distribution**

  1. The gateway ECU generates a session key SSK using HKDF (HMAC based Key Derivation Function): $SSK = HKDF(R_2)$ where R_2 is a 128-bit random number;
  2. The gateway ECU generates a ciphertext $CT_2 = Enc(K_{P2}, SSK)$ using AES encryption, and transmits CT_2 to the PnP device;
  3. The PnP device computes $SSK = Dec(K_{P2}, CT_2)$ using AES decryption and obtains SSK.

5.4 Implementation results

In this section, we describe the result of PoC simulation. For the preparation phase, we evaluated the performance of main algorithms executed by the CA server, the PnP device manufacturer server, and the gateway ECU. The main algorithms include signing of CRL, verification of CRL signature, certificate validation, and secret key generation. CA server, PnP device manufacturer server, and gateway ECU were configured in the same experimental environment.

The experiments were conducted under a PC with processor core i7-4702MQ CPU @ 2.20GHz with 8GB RAM. All the implementations were written in C and the OpenSSL library was used to implement a secure channel between the PnP device manufacturer and the gateway ECU. And the CANoe 8.5 (simulation tool by Vector GmbH) was used to measure the performance of CAN-FD for ECU.

Table 6 shows the average number of executions per second for each algorithm based on 10,000 iterations. Since the CA server and the (PnP device) manufacturer server in the real world can reasonably be inferred to have similar (or much better) performance to the PC used in the experiment, the main algorithms of the Sym-SSL framework can be fully deployed in real-world applications. Likewise, if the gateway ECU has the capability to process public key-based operations in real time, it can sufficiently handle the verification of CRL signature and certificate validation.

Algorithm	Subject	Number of executions per second
Signing of CRL	- CA server	1,330
Verification of CRL signature	- Manufacturer server - Gateway ECU	7,627
Certificate validation	- Manufacturer server - Gateway ECU	3,410
Secret key generation	- Manufacturer server	632

Table 6: Performance evaluation of Sym-SSL framework

For the authentication phase and session key distribution, we performed an experiment by emulating a virtual ECU on CANoe simulator. Therefore, the computing power of the virtual ECU is the same as that of the PC used in the experiment. In the simulation, AES encryption scheme and hash function are used, which are already known to be able to perform within $600\mu s$ by an ECU with 600MHz computing

power [24]. Therefore, it can be fully inferred that symmetric key-based authentication and session key generation are possible in real time in the gateway ECU.

References

[1] D.K. Nilsson, P.H. Phung, and U.E. Larson. Vehicle ecu classification based on safety-security characteristics. In *Proc. of IET Road Transport Information and Control-RTIC 2008 and ITS United Kingdom Members' Conference, Manchester, UK*. IET, July 2008.

[2] F. Kohnhäuser, D. Püllen, and S. Katzenbeisser. Ensuring the safe and secure operation of electronic control units in road vehicles. In *Proc. of the 2019 IEEE Security and Privacy Workshops (SPW'19), San Francisco, CA, USA*, pages 126–131. IEEE, September 2019.

[3] D. Kopencova and R. Rak. Issues of vehicle digital forensics. In *Proc. of the 12th International Science-Technical Conference Automotive Safety, Kielce, Poland*, pages 1–6. IEEE, October 2020.

[4] A. Ghosal and M. Conti. Security issues and challenges in v2x: A survey. *Computer Networks*, 169:107093, 2020.

[5] H. Stoll, D. Grimm, M. Schindewolf, M. Brodatzki, and E. Sax. Dynamic reconfiguration of automotive architectures using a novel plug-and-play approach. In *Proc. of the 2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops'21), Nagoya, Japan*, pages 70–75. IEEE, July 2021.

[6] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, et al. Experimental security analysis of a modern automobile. In *Proc. of the 2010 IEEE symposium on Security and Privacy (IEEE S&P'10), Oakland, CA, USA*, pages 447–462. IEEE, July 2010.

[7] 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 *Proc. of the 20th USENIX Security Symposium (USENIX Security'11), San Francisco, CA, USA*. USENIX, August 2011.

[8] C. Miller and C. Valasek. Adventures in automotive networks and control units. *Def Con*, 21(260-264):15–31, 2013.

[9] C. Miller and C. Valasek. A survey of remote automotive attack surfaces. *Black Hat USA*, 2014:94, 2014.

[10] C. Miller and C. Valasek. Remote exploitation of an unaltered passenger vehicle. *Black Hat USA*, 2015(S 91), 2015.

[11] P. Carsten, T.R. Andel, M. Yampolskiy, and J.T. McDonald. In-vehicle networks: Attacks, vulnerabilities, and proposed solutions. In *Proc. of the 10th Annual Cyber and Information Security Research Conference (CISR'15), Oak Ridge, TN, USA*, pages 1:1–1:8. ACM, April 2015.

[12] A.V. Herrewege, D. Singelee, and I. Verbauwhede. Canauth-a simple, backward compatible broadcast authentication protocol for can bus. In *Proc. of the 2011 ECRYPT workshop on Lightweight Cryptography, Louvain-la-Neuve, Belgium*, volume 2011, page 20, November 2011.

[13] B. Groza, S. Murvay, A.V. Herrewege, and I. Verbauwhede. Libra-can: A lightweight broadcast authentication protocol for controller area networks. In *Proc. of the 11th International Conference on Cryptology and Network Security (CANS'12), Darmstadt, Germany*, volume 7712 of *Lecture Notes in Computer Science*, pages 185–200. Springer, Berlin, Heidelberg, December 2012.

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

[15] 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, 2014.

[16] R. Kurachi, Y. Matsubara, H. Takada, N. Adachi, Y. Miyashita, and S. Horihata. Cacan-centralized authentication system in can (controller area network). In *The 14th International Conference on Embedded Security in Cars (ESCAR'14), Hamburg,Germany*, November 2014.

[17] S. Jain and J. Guajardo. Physical layer group key agreement for automotive controller area networks. In *Proc. of the 2016 International Conference on Cryptographic Hardware and Embedded Systems (CHES'16), Santa Barbara, CA, USA*, volume 9813 of *Lecture Notes in Computer Science*, pages 85–105. Springer, Berlin, Heidelberg, August 2016.

[18] A.-I. Radu and F.D. Garcia. Leia: A lightweight authentication protocol for can. In *Proc. of the 2016 European Symposium on Research in Computer Security (ESORICS'16), Heraklion, Greece*, volume 9879 of *Lecture Notes in Computer Science*, pages 283–300. Springer, Cham, September 2016.

[19] Z. Xiao, F. Li, R. Wu, H. Jiang, Y. Hu, J. Ren, C. Cai, and A. Iyengar. Trajdata: on vehicle trajectory collection with commodity plug-and-play obu devices. *IEEE Internet of Things Journal*, 7(9):9066–9079, 2020.

[20] C. Tymoszek, S.S. Arora, K. Wagner, and A.K. Jain. Dashcam pay: A system for in-vehicle payments using face and voice. arXiv:2004.03756, 2020. `https://doi.org/10.48550/arXiv.2004.03756`.

[21] ISO Central Secretary. IT Security techniques – Entity authentication – Part 2: Mechanisms using authenticated encryption. Standard ISO/IEC 9798-2:2019, International Organization for Standardization, Geneva, CH, 2019.

[22] ISO Central Secretary. IT Security techniques – Entity authentication – Part 3: Mechanisms using digital signature techniques. Standard ISO/IEC 9798-3:2019, International Organization for Standardization, Geneva, CH, 2019.

[23] ISO Central Secretary. IT Security techniques – Key management – Part 2: Mechanisms using symmetric techniques. Standard ISO/IEC 11770-2:2018, International Organization for Standardization, Geneva, CH, 2018.

[24] D. Yu, R.-H. Hsu, and J. Lee. Ec-svc: Secure can bus in-vehicle communications with fine-grained access control based on edge computing. arXiv:2010.14747, 2020. `https://doi.org/10.48550/arXiv.2010.14747`.

Author Biography

Minhye Seo received her B.S. degree in Mathematics and Economics and Ph.D. in information security from Korea University, Seoul, Korea, in 2012 and 2020, respectively. She was a research professor of the institute of cybersecurity & privacy at Korea University. She is currently a assistant professor of the department of Cyber Security at Duksung Women's University, Seoul, Korea. Her research interests include applied cryptography, fuzzy authentication, functional encryption, and lattice-based cryptography.

Appendix

A Entity authentication standards using symmetric techniques (ISO/IEC 9798-2 [21])

In authentication protocols using symmetric-key encryption, it is assumed that two entities participating in the protocol share a secret key. K_{AB} denotes the secret key shared by two entities A and B.

A.1 Unilateral authentication protocols

In unilateral authentication protocols, entity B checks whether entity A is legitimate or not. That is, A is the *prover* and B is the *verifier* in the protocol.

- **One-pass unilateral authentication protocol**

  1. A encrypts both a timestamp T_A and B's identity ID_B with the secret key K_{AB}, and sends the ciphertext C to B;
  2. B decrypts C with K_{AB} to obtain the timestamp T and the identity ID. If T is valid and $ID = ID_B$, then B authenticates A as a legitimate one.

- **Two-pass unilateral authentication protocol**

  1. B selects a random number N_B and sends it to A;
  2. A encrypts both N_B and entity B's identity ID_B with the secret key K_{AB}, and sends the ciphertext C to B;
  3. B decrypts C with K_{AB} to obtain the number N and the identity ID. If $N = N_B$ and $ID = ID_B$, then B authenticates A as a legitimate one.

A.2 Mutual authentication protocols

In mutual authentication protocols, entities A and B authenticate each other.

- **Two-pass mutual authentication protocol**

  1. A encrypts both a timestamp T_A and B's identity ID_B with the secret key K_{AB}, and sends the ciphertext C_A to B;
  2. B encrypts both a timestamp T_B and A's identity ID_A with the secret key K_{AB}, and sends the ciphertext C_B to A;
  3. A (resp. B) decrypts C_B (resp. C_A) with K_{AB} to obtain the timestamp T and the identity ID (resp. the timestamp T' and the identity ID'). If T is valid and $ID = ID_A$ (resp. T' is valid and $ID' = ID_B$), then B (resp. A) authenticates A (resp. B) as a legitimate one.

- **Three-pass mutual authentication protocol**

  1. B selects a random number N_B and sends it to A;
  2. A selects a random number N_A, encrypts N_A, N_B and entity B's identity ID_B with the secret key K_{AB}, and sends the ciphertext C_A to B;
  3. B decrypts C_A with K_{AB} to obtain N_1, N_2 and the identity ID. If $N_2 = N_B$ and $ID = ID_B$, then B authenticates A as a legitimate one. B encrypts N_2 and N_1 with K_{AB} and sends the ciphertext C_B to A;
  4. A decrypts C_B with K_{AB} to obtain N_2' and N_1'. If $N_2' = N_B$ and $N_1' = N_A$, then A authenticates A as a legitimate one.

B Entity authentication standards using asymmetric techniques (ISO/IEC 9798-3 [22])

In authentication protocols using digital signatures, it is assumed that each entity participating in the protocol has his/her own public-private key pair. We denote that the key pair of entity A as (pk_A, sk_A) and the key pair of entity B as (pk_B, sk_B).

B.1 Unilateral authentication protocols

In unilateral authentication protocols, entity B checks whether entity A is legitimate or not. That is, A is the *prover* and B is the *verifier* in the protocol.

- **One-pass unilateral authentication protocol**
  1. A signs both a timestamp T_A and B's identity ID_B with the signing key sk_A, and sends a message (T_A, ID_B) and signature σ to B;
  2. B verifies σ with A's verification key pk_A, and authenticates A as a legitimate one if the verification succeeds.
- **Two-pass unilateral authentication protocol**
  1. B selects a random number N_B and sends it to A;
  2. A selects a random number N_A, signs N_A, N_B, and B's identity ID_B with the signing key sk_A, and sends a message (N_A, N_B, ID_B) and signature σ to B;
  3. B verifies σ with A's verification key pk_A, and authenticates A as a legitimate one if the verification succeeds.

B.2 Mutual authentication protocols

In mutual authentication protocols, entities A and B authenticate each other.

- **Two-pass mutual authentication protocol**
  1. A signs both a timestamp T_A and B's identity ID_B with the signing key sk_A, and sends a message (T_A, ID_B) and signature σ_A to B;
  2. B signs both a timestamp T_B and A's identity ID_A with the signing key sk_B, and sends a message (T_B, ID_A) and signature σ_B to A;
  3. B (resp. A) verifies σ_A (resp. σ_B) with A's verification key pk_A (resp. B's verification key pk_B), and authenticates A (resp. B) as a legitimate one if the verification succeeds.
- **Three-pass mutual authentication protocol**
  1. B selects a random number N_B and sends it to A;
  2. A selects a random number N_A, signs N_A, N_B, and B's identity ID_B with the signing key sk_A, and sends a message (N_A, N_B, ID_B) and signature σ_A to B;
  3. B verifies σ_A with A's verification key pk_A, and authenticates A as a legitimate one if the verification succeeds. B signs N_B, N_A, and A's identity ID_A with the signing key sk_B and sends a message (N_B, N_A, ID_A) and signature σ_B to A;
  4. A verifies σ_B with B's verification key pk_B, and authenticates B as a legitimate one if the verification succeeds.