Received 2 January 2023, accepted 2 March 2023, date of publication 6 March 2023, date of current version 13 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3253626

RESEARCH ARTICLE

TB-Logger: Secure Vehicle Data Logging Method Using Trusted Execution Environment and Blockchain

DONGWOO KANG AND HYO JIN JO
Department of Software, Soongsil University, Seoul 06978, Republic of Korea
Corresponding author: Hyo Jin Jo (hyojin.jo@ssu.ac.kr)

This work was supported by the Institute of Information and Communications Technology Planning and Evaluation (IITP) grant funded by the Korea Government (MSIT) (Robust AI and Distributed Attack Detection for Edge AI Security) under Grant 2021-0-00511.

ABSTRACT With the development of IT technologies, event data recorder (EDR) devices are now installed in modern vehicles to record and analyze vehicle-related events. As data recorded in EDRs began to be used as *conclusive proof* in courts, many researchers turned their to focus on developing methodologies that can protect data recorded in EDRs from data forgery attacks. In general, these existing methods generate verification values for EDR data by using a digital signature algorithm. However, these methods do not provide a way to protect the data stored in an EDR from data forgery attacks in the event of an emergency, such as a car accident, and it is not possible to properly create a verification value for the EDR data due to unexpected and sudden events like a power supply problem. Thus, in this paper, we propose TB-Logger, a novel method that protects EDR data even when there is an emergency situation. TB-Logger relies on the trusted execution environment (TEE) to protect EDR data from data forgery attacks, which ultimately result in data modification, reordering, and deletion. In addition, in the event of an emergency, TB-Logger utilizes a blockchain system to store verification values and publicly verify the data generated during the event. We evaluated the practicality of TB-Logger using two real vehicles: the Hyundai Avante CN7 and the Tesla Model 3. Through these tests, we confirmed that TB-Logger can generate verification values for EDR data without incurring any data loss.

INDEX TERMS Event data recorder (EDR), trusted execution environment (TEE), blockchain, secure logging.

I. INTRODUCTION

Recently, event data recorder (EDR) devices mounted on vehicles are used to record accidents and analyze accident records. EDR technology exists for the sole purpose of recording and storing crash-related parameters and information shortly before, during, and immediately after a vehicular collision [1]. In particular, EDR technology that is specialized for vehicles is called motor vehicle event data recorder (MVEDR) [2], and many manufacturers now install these EDRs in their current model products [3], [4], [5].

The associate editor coordinating the review of this manuscript and approving it for publication was Yan Huo.

An EDR records at least 15 parameters (e.g., time, speed, safety belt status, etc.) specified by the U.S. National Highway Traffic Safety Administration (NHTSA) as well as safety-related information, such as the vehicle's current speed and brake status. Data to be recorded in the EDR is generated by electronic control units (ECU) and then transmitted to the EDR over an in-vehicle network, such as the controller area network (CAN).

Since data stored in the EDR began to be used as *conclusive proof* in courts [6], there are new concerns in the industry regarding the possibility of data forgery attacks (e.g., data tampering, deletion) that could damage the information recorded in an EDR [7]. In light of this, some research produced a digital signature-based logging method for the

TABLE 1. Data stored in an EDR 30 seconds before and after an accident occurs.

	Pre-event message	Post-event message
Parameters	• UTC-time of event • Brake switch (on/off) • Clutch switch (on/off) • Accelerator pedal position (%) • Vehicle speed (MPH) • Engine speed (RPM) • Crash status (airbag/seatbelt/indicators/etc.) • Acceleration/deceleration value (that triggered incident event)	• UTC-time of event • Brake switch (on/off) • Clutch switch (on/off) • Accelerator pedal position (%) • Vehicle speed (MPH) • Engine speed (RPM) • Crash status (airbag/seatbelt/indicators/etc.)

EDR [8], but the solution proved not to be cost-effective in terms of data processing because the EDR would need to generate verification values for hundreds of CAN messages per second.

To deal with the limitations of digital signature-based logging techniques, other researchers proposed trusted execution environment (TEE)-based methods that leveraged both a digital signature algorithm and a message authentication code (MAC) algorithm [9], [10]. These TEE-based works first define a basic data unit (e.g., a CAN message) and a basic data block (e.g., a specific number of consecutive CAN messages). Then, the TEE-enabled EDR generates a MAC value for each basic data unit. When the number of basic data units collected equals the number defined by the logging system, the basic data units create a basic data block, and the TEE-enabled EDR generates a digital signature value on it. However, despite any potential efficacy of this method, it cannot handle emergency situations in which a basic data block without its corresponding TEE digital signature value remains unprocessed due to a power supply failure or physical damage to the TEE module caused by an event like a car accident. Therefore, this present study proposes a new type of TEE and blockchain-based EDR named TB-Logger that is capable of protecting EDR data even when an emergency occurs.

In the proposed system, an authentication key (AK) initialized by TEE is divided into n partial secret values through Shamir's (t, n) secret sharing [11]. Then, these partial secret values are distributed to n key-restoring participants over a secure channel, such as the transport layer security (TLS). The key-restoring participants can be vehicle owners, original equipment manufacturers (OEMs), insurance companies, police offices, courts, or other qualifying parties. To support secure data logging, the TEE module installed on TB-Logger generates MAC values for all basic data units using the AK. When the number of basic data units reaches a specific threshold, the data units are regarded as one basic data block and the TEE module generates a digital signature value on that block. Additionally, when verification of EDR data is required, data verification is performed. If there is no emergency and every basic data block has its own valid digital signature value (i.e., a complete data block), the integrity of each data block is verified via the corresponding digital signature value. Otherwise, if an incomplete data block is missing a digital signature value, the basic data units in the incomplete block are uploaded to a blockchain system and then key-restoring participants more than $t(1 \leq t \leq n)$ restore the secret value (i.e., AK) to verify the MAC values of the basic data units in that specific incomplete data block.

The contribution of this paper is as follows.

1) There are numerous data protection mechanisms for EDR. These protection methods are usually based on a MAC algorithm and a digital signature algorithm to provide security goals such as integrity, authentication, and non-repudiation for EDR data. In general, existing methods create a basic data block after a certain number of basic data units have been collected and generate a digital signature value for that block. However, there is no method capable of functioning during an emergency in which the EDR system fails to generate a digital signature on a basic data block due to abnormal circumstances, such as a car accident. To our knowledge, we introduce for the first time a technique called TB-Logger that utilizes both TEE and blockchain systems to verify logging data even when an emergency arises.
2) We verified the performance of TB-Logger in two real vehicles: the Hyundai Avante CN7 and the Tesla Model 3. Through our experiment, we confirmed that, if the TB-Logger adopts RSA 2048 as a digital signature algorithm and HMACwithSHA256 as a MAC algorithm, data loss is 0% when the block size is greater than 1,687 for the Hyundai Avante CN7 and 5,657 for the Tesla Model 3.

The composition of this paper is as follows. Section II describes the background and related works of this paper. We explain our system model and the proposed system, TB-Logger, in Section III and IV, respectively. Section V gives an overview of the security and performance evaluation of our system. Finally, Section VI concludes this paper.

II. BACKGROUND AND RELATED WORKS

A. BACKGROUND

1) EVENT DATA RECORDER

The event data recorder (EDR) was originally developed to record and analyze accidents in airplanes, but recently, manufacturers have adapted this technology for vehicles. In particular, the EDR mounted on a vehicle is specifically referred to as the motor vehicle event data recorder (MVEDR) [2].

EDR technologies are mainly developed by vehicle manufacturers and mounted during the manufacturing of vehicles [12], [13]. These devices are capable of capturing low-bandwidth data including but not limited to vehicle speed, engine speed, throttle, brake status, and acceleration [14]. For example, if the speed of the vehicle is rapidly decreasing, EDR collects data as shown in Table 1 [15]. As of November 2020, the American Automobile Association (AAA) reported that approximately 95 percent of vehicles in the United States were already equipped with EDRs [16], thus highlighting the need for a method that can protect the data recorded by these devices.

2) CONTROLLER AREA NETWORK

Robert Bosch developed the controller area network (CAN) in 1986 and it later became an industry standard. CAN is a physical structure in which two copper wires are twisted to create a pair. This structure is resistant to electrical noise and supports up to 1 Mbps of information transmission, but actual vehicles mainly use a speed of 500 Kbps. CAN transmits information in a frame up to 8 bytes max and consists of bus-type network topology [17].

Data transmitted over the CAN includes information on vehicle operation and safety, such as current vehicle speed, seat belt engagement, brake status, engine status, and so on.

3) TRUSTED EXECUTION ENVIRONMENT

The trusted execution environment (TEE) is a secure area of the main processor. It ensures that the code and data loaded internally are protected in terms of confidentiality and integrity. The TEE provides security functions such as isolated execution, the integrity of applications, and the confidentiality of information assets. Representative hardware technologies that can be used to support TEE implementations include ARM TrustZone, Intel-SGX, and AMD Secure Execution Environment.

Among these, ARM TrustZone is mainly found in embedded systems and provides an environment separated into the rich execution environment (REE), a general execution area, and the TEE, a secured execution area. The REE is controlled by general OS and is an environment in which applications that do not need secure areas are separated from attack surfaces such as network interfaces. On the other hand, to provide some secure areas, the TEE is configured based on a secure OS that offers isolated and protected resources for security-sensitive applications [18].

4) SHAMIR's (t, n) SECRET SHARING

Shamir's (t, n) secret sharing ((t, n)-SSS) is based on polynomial computation and Lagrange's interpolation. In (t, n)-SSS, there are n participants and a trusted dealer. The n partial secret, $f(x_t)$, is generated via Eq. 1 after random values (x_t) are selected, where $1 \leq t \leq n$. Then, a trusted dealer distributes these partial secret values to participants. The secret value (i.e., $f(0) = S$) can be reconstructed based on the Lagrange interpolation polynomial by way of Eq. 2 taking any t partial secret values of participants. When there are fewer than t participants, the original secret cannot be reconstructed.

$$f(x) = S + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1} \tag{1}$$

$$f(x) = \sum_{j=1}^{t} \left(y_j \times \prod_{1 \leq o \leq t, o \neq j} \frac{x - x_o}{x_j - x_o} \right) \tag{2}$$

5) BLOCKCHAIN

Blockchain is a shared, immutable ledger that facilitates the process of recording transactions and tracking assets. Note that the blockchain nodes are controlled by a peer-to-peer (P2P) network to maintain the blockchain system. In the blockchain system, users can record and review data, but they cannot modify or delete any pre-existing data. A consensus algorithm—such as Proof-of-Work (PoW), Proof-of-Stake (PoS), and Practical Byzantine Fault Tolerance (PBFT)—is used to reach a common agreement about the present state of data stored in the blockchain.

B. RELATED WORKS

There are many techniques to protect logging data from data modification attacks. In general, protection methods for logging data can be classified into two types: a non-TEE-based protection approach and a TEE-based approach.

1) NON-TEE-BASED PROTECTION APPROACH

Bellare et al. proposed a logging system based on a cryptographic hash function to detect manipulation of logging data [19], [20]. In these works, a chain of secret keys generates message authentication code (MAC) values for every pre-defined time interval using the hash-chain method. Likewise, Schneier and Kelsey designed a MAC-based logging system to deal with data manipulation attacks on logging data [21]. Recently, blockchain-based data logging studies have been actively conducted for healthcare services. In the works [22], [23], [24], [25], [26], logging systems were proposed to collect data from healthcare IoT devices and to safely record data in the blockchain. In addition, the works [27] and [28] provide blockchain-based data management methods considering General Data Protection Regulation (GDPR).

However, these non-TEE-based protection methods have a key management problem, the secret values used to check for data manipulation are stored in untrusted environments where an attacker could compromise and exploit the system.

2) TEE-BASED PROTECTION APPROACH

Lee et al. proposed a secure logging method based on ARM TrustZone, which enables the existing logging system (i.e., `Syslog`) to generate secure logs through inter-process communications efficiently [29]. However, since ARM TrustZone cannot be directly applied to the EDR system, researchers developed T-Box to provide a secure logging environment for the EDR system [9]. In this system, a two-dimensional hash chain is used to ensure the integrity and continuity of logging data. Recently, an attribute-based encryption method was proposed to support fine-grained access control of EDR data [10]. In addition, Lu et al. proposed TEE-based SQLite for embedded systems [30]. The system utilizes TEE to encrypt sensitive data and safely perform sensitive operations by isolating them from unreliable environments. However, none of these existing TEE-based works can protect logging data in emergency situations like a car accident wherein the EDR system fails, resulting in the verification failure of logged data. Therefore, we propose a new TEE-based logging method called TB-Logger, which can verify logging data even during such emergency situations.

III. SYSTEM MODEL

A. MOTIVATION

According to [31], if EDR data is exposed to an attacker without any protection method, the EDR data can be arbitrarily modified by the attacker. In order for the EDR data to be used as essential information for identifying the root cause of an emergency situation like a vehicle accident, data generated in the emergency situation must not be manipulated by an entity with malicious intent. Although there are several methodologies that can protect the EDR data from data forgery attacks, they do not deal with data verification issues that may occur when the EDR system fails to fully operate due to emergency situations (e.g., power supply issues caused by vehicle accidents). Therefore, there is a need for a study that can verify the EDR data even if the EDR system fails in an emergency.

B. THREAT MODEL

In the EDR system, an attacker can perform a data forgery attack on the logging data. For example, in the event of a vehicle collision, the driver may modify or delete data recorded in the EDR system to conceal a driving mistake, leading to the untrustworthiness of the system. Contrary to the above-described case, if a vehicle collisions due to defects in the vehicle, the OEM may also modify or delete data recorded in the EDR to hide their manufacturing fault. This is also an undesirable situation. In light of this, the EDR system must be secured against data forgery attacks with a robust data verification method that can check the integrity of the EDR data even when an emergency event arises, such as a car accident.

C. SYSTEM OVERVIEW

The proposed architecture of TB-Logger is shown in Fig. 1. TB-Logger consists of four steps: 1) (t, n)-Key Initialization, 2) Secure Logging, 3) Preparation of Logging Data Verification, and 4) Logging Data Verification.

- (t, n)-Key Initialization: The AK generated by the TEE is divided into n partial secret values through the (t, n)-SSS scheme, and these are distributed to n key distribution targets through secure channels such as the TLS. Key distribution targets may be vehicle owners, vehicle manufacturers, insurance companies, police offices, or courts. These distribution targets are called key-restoring participants. Since a partial secret value cannot be used alone for AK restoration, successful key restoration can be performed only when more than t partial secret values have been collected.
- Secure Logging: For each basic data unit recorded in the EDR system, a MAC is generated using the AK protected by the TEE. When a predefined number of basic data units is recorded or a predefined timer expires, a basic data block is formed and then signed by the TEE private key, $Priv_{TEE}$. If there is no TB-Logger failure caused by an emergency, and all basic data blocks have their own digital signature values, they are labeled complete data blocks, or $Block_{complete}$. Otherwise, the last basic data block recorded in TB-Logger will not have a legitimate signature value due to a TB-Logger failure caused by an emergency. In this case, the last basic data block is labeled an incomplete data block, or $Block_{incomplete}$.
- Preparation of Logging Data Verification: When data verification is required, TB-Logger performs the data verification preparation step by dividing it into two cases: 1) Preparation without incomplete data blocks and 2) Preparation with an incomplete data block. If all basic data blocks are complete, they are directly transferred to the next step, Logging Data Verification, along with their corresponding digital signature values. Otherwise, if there is an incomplete data block, the cryptographic hash value of this block is transferred to the blockchain system.
- Logging Data Verification: The data verification steps in TB-Logger consist of: 1) Verification with complete data blocks and 2) Verification with an incomplete data block. When there are complete data blocks, verification of basic data blocks in TB-Logger is performed through the TEE public key, Pub_{TEE}. On the other hand, when there is an incomplete data block, more than t key-restoring participants have to restore the AK used to generate the MAC values for the basic data units. The restored key is then used to perform MAC verification for the basic data units in the incomplete data block.

D. ASSUMPTIONS

When building a TEE environment in an EDR system, we assume that it is difficult for an attacker to compromise

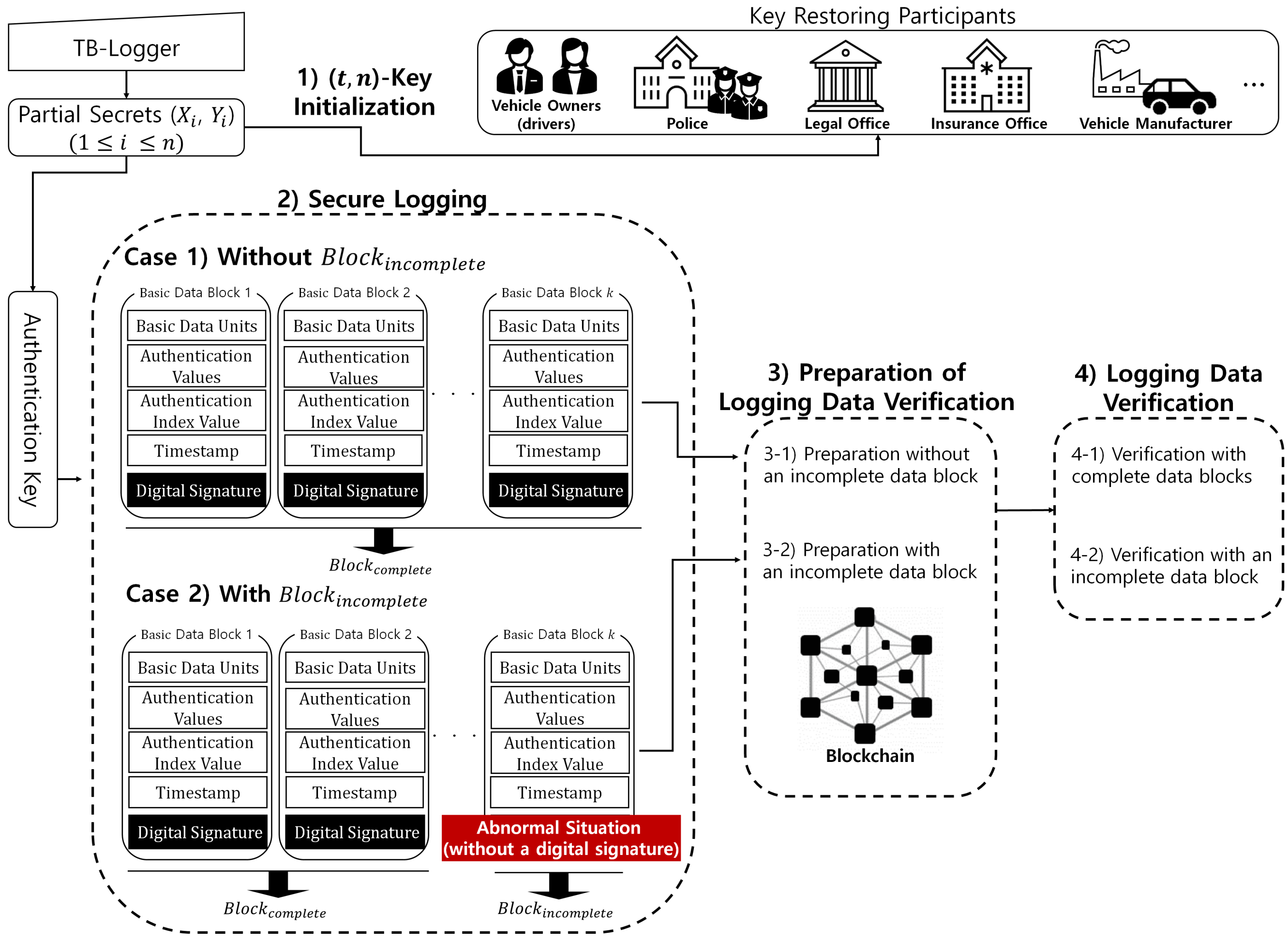
TB-Logger
Partial Secrets (X_i, Y_i) ($1 \le i \le n$)
1) (t, n)-Key Initialization
Key Restoring Participants
Vehicle Owners (drivers)
Police
Legal Office
Insurance Office
Vehicle Manufacturer
2) Secure Logging
Authentication Key
Case 1) Without $Block_{incomplete}$
Basic Data Block 1
Basic Data Block 2
Basic Data Block k
Basic Data Units
Authentication Values
Authentication Index Value
Timestamp
Digital Signature
$Block_{complete}$
Case 2) With $Block_{incomplete}$
Abnormal Situation (without a digital signature)
$Block_{incomplete}$
3) Preparation of Logging Data Verification
3-1) Preparation without an incomplete data block
3-2) Preparation with an incomplete data block
Blockchain
4) Logging Data Verification
4-1) Verification with complete data blocks
4-2) Verification with an incomplete data block

FIGURE 1. The architecture of TB-Logger.

the TEE environment. For example, the secret values such as the authentication key (AK) and the private key ($Priv_{TEE}$) that are generated by TEE are assumed to be protected from unauthorized access.[1]

In addition, we assume that the key-restoring participants can receive their own partial secret value from the TEE module of an EDR through a secure channel, such as the TLS. Furthermore, key-restoring participants are assumed to become verifiers who take charge of the verification of EDR data. Finally, more than t participants are assumed to not collude for data forgery.

IV. TB-LOGGER

A. (t, n)-KEY INITIALIZATION

(t, n)-key initialization is described as follows.

1) TB-Logger generates a public key (Pub_{TEE}) and a private key ($Priv_{TEE}$) pair as well as an authentication key (AK).
2) Eq. 1 creates a (t, n) secure-sharing function, $f()$. This function is a $(t-1)$th order equation and is used to generate the partial secret values.
3) Then, TB-Logger generates n partial secret values, which are composed of random values x_i and their corresponding values $f(x_i)$, where $1 \le i \le n$. After this, using a secure channel such as the TLS, these values are distributed to n key-restoring participants, such as the vehicle owner, the vehicle manufacturer, the insurance company, an automotive-related administrative agency, or a law enforcement agency.

B. SECURE LOGGING

When TB-Logger records data, it generates MAC values $Auth_{Data_{i,j}}$ on every basic data unit $Data_{i,j}$, which is j-th basic data unit in the i-th basic data block.

$$Auth_{Data_{i,j}} = MAC_{AK}(i||j||Data_{i,j}) \tag{3}$$

Then, it updates the authenticated index value $Auth_{index_{i,j}}$ whenever $Auth_{Data_{i,j}}$ is generated as follows.

$$Auth_{index_{i,j}} = MAC_{AK}(time_{i,j}||i||j||Auth_{index_{i,j-1}}) \tag{4}$$

[1]Even though [32] and [33] demonstrated possible attacks on the TEE, we consider these out of the scope of this paper because these attacks can be dealt with using other studies [34], [35].

TABLE 2. The notations of TB-Logger.

Notation	Description
t	A threshold required for restoring a secret key
n	The number of key-restoring participants
$Data_{i,j}$	A j-th basic data unit of the i-th basic data block
w	The number of basic data units in a basic data block (i.e., a window size)
$Data_i^w$	A set of basic data units in i-th basic data block
AK	An authentication key
$Auth_{index_{i,j}}$	The authentication index value of $Data_{i,j}$
$Auth_{Data_{i,j}}$	The authentication value of $Data_{i,j}$
$Auth_i^w$	A set of authentication values in i-th basic data block
$time_i$	Timestamp of the i-th basic data block
$time_{i,j}$	Timestamp of $Data_{i,j}$
Pub_{TEE}	The public key of TEE
$Priv_{TEE}$	The private key of TEE
$Block_{complete}$	A complete data block
$Block_{incomplete}$	An incomplete data block
$Hash()$	A cryptographic hash function
$MAC_x()$	A message authentication code using x

where $Auth_{index_{i,j}} = 0$,

$$time_i = time_{i,j} \tag{5}$$

When the total number of $Auth_{Data_{i,j}}$ reaches w, w pairs of $Auth_{Data_{i,j}}$ and $Data_{i,j}$ and the authenticated index $Auth_{index_{i,j}}$ constitutes the basic data block i. Then, the digital signature value for the basic data block i is generated using the TEE private key $Priv_{TEE}$ as follows:

$$Data_i^w = Data_{i,1}||\ldots||Data_{i,w} \tag{6}$$

$$Auth_i^w = Auth_{Data_{i,1}}||\ldots||Auth_{Data_{i,w}} \tag{7}$$

$$h = Hash(time_i||Auth_i||Auth_{index_{i,w}}) \tag{8}$$

$$sv = Sign_{Priv_{TEE}}(h) \tag{9}$$

After generating the signature value sv, TB-Logger stores a complete data block $Block_{complete}$, which is composed of $time_i$, $Data_i^w$, $Auth_i^w$, $Auth_{index_{i,w}}$, and sv. If $time_i$ is not updated for a predefined timer, TB-Logger forms data block i, even though the total number of unprocessed basic data units is less than w.

In an emergency situation such as a car accident, this could potentially cause TB-Logger to fail. To prevent such a failure, the last data block in TB-Logger does not get a valid signature value and an incomplete data block is generated; $Block_{incomplete}$ is composed of $Data_i^j$, $Auth_i^j$, and $Auth_{index_{i,j}}$ where j is less than w.

C. PREPARATION OF LOGGING DATA VERIFICATION

When data verification is required, TB-Logger prepares by first distinguishing between two cases: 1) Preparation without an incomplete data block and 2) Preparation with an incomplete data block.

1) PREPARATION WITHOUT AN INCOMPLETE DATA BLOCK

If there is no incomplete data block, all basic data blocks recorded in TB-Logger have legitimate TEE digital signatures, and the TEE public key allows all basic data blocks to be verified. Thus, all complete data blocks with their corresponding digital signature values are directly transferred to the next step of Logging Data Verification without any additional blockchain logging process.

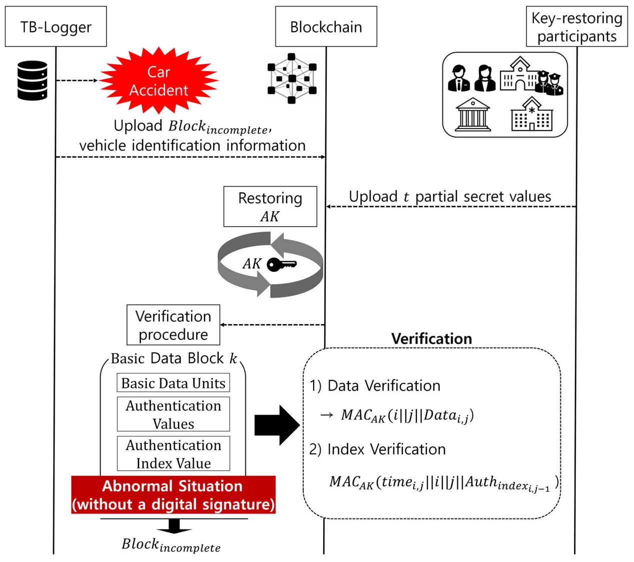
TB-Logger
Blockchain
Key-restoring participants
Car Accident
Upload $Block_{incomplete}$, vehicle identification information
Restoring AK
Upload t partial secret values
AK
Verification procedure
Verification
Basic Data Block k
Basic Data Units
Authentication Values
Authentication Index Value
Abnormal Situation (without a digital signature)
$Block_{incomplete}$
1) Data Verification
$\rightarrow MAC_{AK}(i||j||Data_{i,j})$
2) Index Verification
$MAC_{AK}(time_{i,j}||i||j||Auth_{index_{i,j-1}})$

FIGURE 2. Preparation/verification for the logging data verification step for an incomplete block.

2) PREPARATION WITH AN INCOMPLETE DATA BLOCK

If an incomplete data block exists, there is an additional blockchain logging process that must happen, in which the incomplete data block that lacks a digital signature value is stored in the blockchain system[2] as follows:

1) A verifier computes a hash value of $Block_{incomplete}$, (i.e., $h_{incomplete} = Hash(Block_{incomplete})$). It then sends $h_{incomplete}$ and the corresponding vehicle information number (VIN) to the TEE address of the blockchain system. In general, the TEE public key is used to derive the TEE blockchain address [36]. Then, the verifier sends a key-restoring request to n key-restoring participants.
2) Note that if a key-restoring participant P_i wants to take part in the key-restoring process, it sends its own partial secret value $f(x_{P_i})$ to the TEE blockchain address.

Fig. 2 depicts the detailed process to prepare for the Logging Data Verification step.

D. LOGGING DATA VERIFICATION

Algorithm 1 shows how data recorded in the TB-Logger is processed for verification. Data block verification follows different processing procedures depending on if it is a complete block or an incomplete block.

1) VERIFICATION WITH COMPLETE DATA BLOCKS

When there are complete data blocks, all legitimately signed blocks are verified using the TEE public key Pub_{TEE}.

[2]The blockchain system can be implemented in both permissioned and permissionless blockchains depending on the requirements of the system designer and users.

Algorithm 1 Logging Data Verification

```
Ensure: Success or Fail
     /* Verification with incomplete data blocks */
 if Request is verifying $Auth_{Data_{i,j}}$ then
     /* Verify $MAC_{AK}(i||j||Data_{i,j})$
     and $MAC_{AK}(time_{i,j}||i||j||Auth_{index_{i,j-1}})$ */
     $S \leftarrow t$ partial secret values
     data $\leftarrow i||j||Data_{i,j}$
     index $\leftarrow time_{i,j}||i||j||Auth_{index_{i,j-1}}$
     if $MAC_{AK}$(data) $= MAC_S$(data) and
        $MAC_{AK}$(index) $= MAC_S$(index) then
          return Success
     else
          return Fail
     end if

     /* Verification with complete data blocks */
 else if Request is verifying $Sign_{Priv_{TEE}}(h)$ then
     /* Verify Signature */
     $S \leftarrow t$ partial secret values
     index $\leftarrow time_{i,j}||i||j||Auth_{index_{i,j-1}}$
     ver $\leftarrow$ Verify($Pub_{TEE}$, $Sign_{Priv_{TEE}}(h)$)
     if $MAC_{AK}$(index) $= MAC_S$(index) and
        ver $= 1$ then
          return Success
     else
          return Fail
     end if
 end if
```

2) VERIFICATION WITH AN INCOMPLETE DATA BLOCK

Since there is no valid digital signature value on an incomplete data block, the verifier restores the AK, which is used for verification of $Auth_{Data_{i,j}}$ in the incomplete data block. If there are more than t partial secrets ($f(x_{P_i})$ and x_{P_i}) in the TEE blockchain address, the verifier can recover the AK through the (t, n)-SSS scheme. In addition, $Auth_{index_{i,w}}$ is verified by the AK to confirm that the last block index of data did indeed exist in the incomplete data block.

V. ANALYSIS

In this section, we analyze the security and performance of TB-Logger.

A. SECURITY ANALYSIS

It is a concern that an attacker could modify, reorder, or delete the data in TB-Logger to eliminate or conceal their mistakes. This section explains why TB-Logger is secure against such data forgery attacks like data modification attacks, data reordering attacks, and data deletion attacks.

1) DATA MODIFICATION ATTACK

An attacker could also intentionally modify basic data units in a complete data block or in an incomplete data block.

a: DATA MODIFICATION ATTACKS ON A COMPLETE DATA BLOCK

TB-Logger is based on a digital signature algorithm such as RSA or ECDSA, both of which have been proven to be secure against signature forgery attacks. Since a complete data block has a valid signature, an attacker who wants to modify the basic data unit must forge the corresponding signature value. However, this contradicts the security proof of standardized digital signature algorithms.

b: DATA MODIFICATION ATTACKS ON AN INCOMPLETE DATA BLOCK

An incomplete data block does not have a valid signature value. Therefore, for a data modification attack, the attacker must forge the MAC value ($Auth_{i,j} = MAC_{AK}(i||j||Data_{i,j})$) of a basic data unit or obtain the authentication key (AK) used to generate the MAC. The proposed TB-Logger adopts a hash-based message authentication code (HMAC) to protect the incomplete data block from the attacker. The security strength of HMAC depends on the key size and the output size. According to [37], an n-bit HMAC can be broken after $O(2^{\frac{n}{2}})$ queries to the n-bit MAC generation oracle. This is known as the birthday limit. However, if HMAC-SHA256 is used, the generation of $O(2^{128})$ queries to HMAC-SHA256 is not practical within the polynomial time. This means it is impossible for an attacker, who can access the MAC generation function provided by TB-Logger, to compromise HMAC. The acquisition of the AK by an attacker also contradicts the assumptions of the (t, n)-SSS scheme because obtaining the AK is required for more than t key-restoring participants to be colluded with. Thus, when 128-bit AK is used, the only option for the attacker is to guess the correct 128-bit key out of 2^{128} candidate keys.

In addition, the AK that verifies basic data units can be disclosed via the blockchain system after the AK restoring process by more than t key restoring participants. Thus, if an attacker were to find the AK from the blockchain system, he/she can generate a new MAC value for the forged basic data unit. However, in this case, the attacker must also modify $Hash(Block_{incomplete})$ recorded in the blockchain or find the collision of $Hash(Block_{incomplete})$. Modifying the information recorded in the blockchain system or finding the collision of the cryptographic hash function contradicts the system assumptions of the blockchain system as well as the cryptographic hash function.

2) DATA REORDERING ATTACK

An attacker can change the order of the basic data blocks or change the order of the basic data units within a block. However, if there is a complete data block, a digital signature value would have been calculated along with the timestamp $time_i$, thus rendering it impossible to change the block order. In the case of an incomplete data block, there is no digital signature value, but the order of basic data units can still be checked through the index values i and j, which are included in the MAC value $Auth_{i,j} = MAC_{AK}(i||j||Data_{i,j})$. Thus, reordering attacks on an incomplete data block also becomes impossible.

3) DATA DELETION ATTACK

An attacker may attempt to hide the information recorded by TB-Logger by deleting the basic data blocks or the basic data units. However, deletion of a complete data block when using TB-Logger would be detected through the timestamp of the basic data block $time_i$. If the last basic data block related to an accident is completely deleted, then the block deletion can be detected because there would be no block with the timestamp relating to the accident.

In addition to the detection of block deletion, TB-Logger can also detect the deletion of basic data units in an incomplete data block using $Auth_{index_{i,j}}$ because $Auth_{index_{i,j}}$ is updated with a new value each time a new basic data unit and its MAC value are generated. For example, when the j-th basic data unit and its MAC value are generated, $Auth_{index_{i,j}}$ is generated and $Auth_{index_{i,j-1}}$ value is deleted. Therefore, if an attacker wished to delete the basic data unit j, he/she would have to regenerate $Auth_{index_{i,j-1}}$, which is impossible without obtaining the AK protected by the TEE.

B. PERFORMANCE EVALUATION

1) EVALUATION ENVIRONMENT

To conduct the performance evaluation of TB-Logger, we used Raspberry Pi Model 3B with Open Portable Trusted Execution Environment (OP-TEE) [38] as OP-TEE supports a secure operating system and standardized GlobalPlatform TEE libraries. Table 3 shows the implementation environment of TB-Logger.

The states of the CAN bus are divided into driving and idle. In general, a lot of CAN data is generated in driving states accordingly, this experiment was conducted using CAN data obtained in driving states. In our experiments, TB-Logger uses CAN packets collected from the Hyundai Avante CN7 and the Tesla Model 3 on the driving state in urban areas.

2) PERFORMANCE EVALUATION

To evaluate the performance of TB-Logger, we implemented the basic operations of TB-Logger, which are MAC generation/verification and digital signature generation/verification. We obtained the average execution time shown in Table 4 after executing each operation 100 times. Likewise, TEE's world switching between the normal world and the secure world was performed 100 times, and T_{ws} was confirmed to be 90 us on average.

Generally speaking, data recording in TB-Logger should be faster than block generation time $T_{datagen} = time_i - time_{i-1}$. In TB-Logger, a basic data block consists of up to w basic data units. Since two MAC operation times (T_{mac}) need to be performed for each basic data unit, TB-Logger must perform $2 \times w \times T_{mac}$. Additionally, at this point, we also need the digital signature generation time for the basic data block T_{sign} and the world switching time T_{ws} between the secure world and the normal world. Based on our observations, TB-Logger must satisfy the following Eq. 10.

$$T_{datagen} \geq 2 \times w \times T_{mac} + T_{sign} + T_{ws} \tag{10}$$

If TB-Logger violates Eq. 10, data loss occurs as the basic data block recording time would take longer than the basic data block generation time. The data loss rate can be calculated using Eq. 11:

$$data\ loss = \frac{N_{gen} - N_{rec}}{N_{gen}}, \tag{11}$$

where N_{gen} and N_{rec} are the total number of generated and recorded basic data blocks, respectively.

To evaluate the practicality of the TB-Logger, the data loss rate was measured using two experimental vehicles, the Hyundai Avante CN7 and the Tesla Model 3. Fig. 3 shows the number of basic data units generated by these two experimental vehicles, in which the CAN bandwidth is 500 kbps. In the case of the Hyundai Avante CN7, it generated 12,000 to 13,000 packets per second, with an average of 12,628 packets. In the case of the Tesla Model 3, it generated 14,000 to 15,000 packets per second, with an average of 14,471 packets. Fig. 4 shows the data loss rate according to the size of the basic data block for each vehicle. In our experiments with the Hyundai Avante CN7, the data loss rate is 0% when the basic data block size is greater than 907 (RSA1024) or 1,687 (RSA2048). In the case of the Tesla Model 3, the data loss rate is 0% when the basic data block size is greater than 3,041 (RSA1024) or 5,657 (RSA2048).

3) HANDLING EMERGENCY CASES

To evaluate how TB-Logger deals with emergency cases such as power supply issues caused by a car accident, we randomly shut down the TB-Logger implemented on Raspberry

TABLE 3. Specifications of our evaluation environment.

Category	Specifications
SoC	Broadcom BCM2837 chipset
CPU	ARM CortexTM-A53 Quad-core 64-bit (ARMv8)
Frequency	1.2GHz
RAM	1GB LPDDR2-900 SDRAM
Storage	8GB eMMC
Software	OP-TEE v3.14.0

TABLE 4. Implementation results on raspberry Pi model 3B with OP-TEE.

MAC Generation (32KB)		MAC Verification (32KB)	
Type	Execution time (us)	Type	Execution time (us)
HMAC-SHA1	61	HMAC-SHA1	61
HMAC-SHA224	62	HMAC-SHA224	64
HMAC-SHA256	63	HMAC-SHA256	67
HMAC-SHA384	82	HMAC-SHA384	82
HMAC-SHA512	80	HMAC-SHA512	81
Signing		**Verifying**	
Type	Execution time (ms)	Type	Execution time (ms)
RSA1024	301	RSA1024	11
RSA2048	560	RSA2048	13

TABLE 5. The comparison between TB-Logger and the existing works.

	Approach	Data Continuity	Key protection	Data verification in case of emergency	Data forgery detection in case of emergency
Non-TEE-based protection	M. Bellare et al. [19], [20]	✓	✗	✗	✗
	B. Schneier et al. [21]	✓	✗	✗	✗
	Blockchain-based [22]–[26]	✓	✗	✗	✗
TEE-based protection	S. Lee et al. [9], [10], [29]	✓	✓	✗	✗
	D. Lu et al. [30]	✓	✓	✗	✗
	TB-Logger (Proposed Approach)	✓	✓	✓	✓

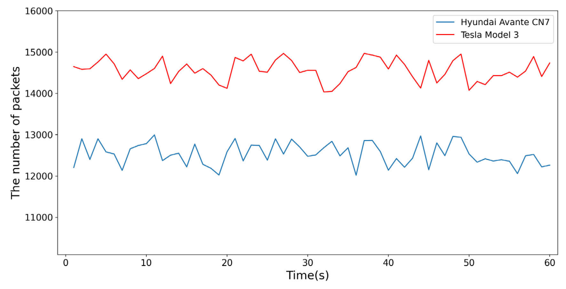

FIGURE 3. The number of CAN packets per time window on the driving state of the Hyundai Avante CN7 and the Tesla model 3.

Pi Model 3B while TB-Logger is running. It was confirmed that only $Block_{complete}$s are stored when the TB-Logger is operating normally. However, when TB-Logger is turned off, it was confirmed that the TB-Logger stored the $Block_{incomplete}$ where the digital signature for the corresponding block does not exist.

To deal with this worst-case where $Block_{incomplete}$ exists, TB-Logger is designed to upload $Block_{incomplete}$ to the blockchain before restoring the AK by key restoring participants. Then, the AK is restored through Shamir's (t, n) secret sharing and is used for verifying $Auth_{Data_{i,j}} = MAC_{AK}(i||j||Data_{i,j})$ to validate whether $Data_{i,j}$ in $Block_{incomplete}$ is corrupted.

C. COMPARISON

Table 5 shows the comparison between TB-Logger and the existing works. Non-TEE-based protection works (e.g., [19], [20], [21]) ensure data continuity in the logging system, but they cannot provide key protection. In addition, they cannot provide data verification and data forgery detection in case of emergency. Otherwise, TEE-based protection works (e.g., [9], [10], [29]) ensure data continuity and provide the key protection. However, the existing TEE-based works still cannot provide the data verification and data forgery detection in case of emergency. Unlike the existing works, TB-Logger can provide data verification and data forgery detection in case of emergency as well as data continuity and key protection.

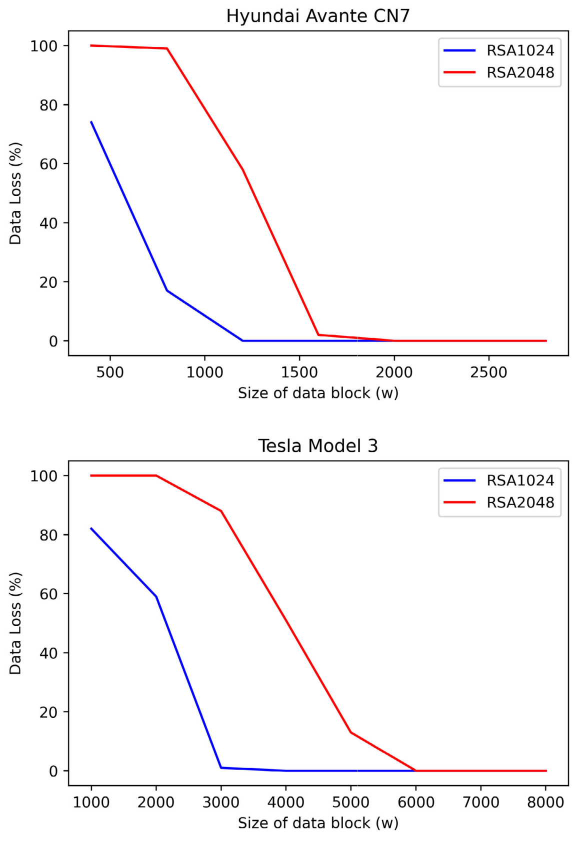

FIGURE 4. Data loss rate by block size.

VI. DISCUSSION

A. EXPERIMENTS ON EMERGENCY CASES

In order to measure the overhead of TB-Logger, two real vehicles (Hyundai Avante CN7 and Tesla Model 3) were used. However, for safety reasons, it was not possible to create an emergency situation such as a vehicle accident in the TB-Logger experiment. To overcome the experimental limitations related to these emergency situations, we conducted an experiment in which the power of the TB logger (i.e., OP-TEE enabled Raspberry Pi 3B) was arbitrarily shut down. As a result of the experiment, we found $Block_{incomplete}$

that cannot be verified in the existing techniques but can be verified in the proposed technique.

B. TEE's VULNERABILITIES

According to the works [32] and [33], cyber attacks on TEE have been introduced. Since TB-Logger is based on the assumption that TEE is not compromised, vulnerabilities in the TEE can lead to problems with the TB logger's data protection. However, TEE's vulnerabilities could be dealt with the existing countermeasures [34], [35]. In the future, it is expected that the attack surfaces on TEE will be decreased by an advanced version of TEE.

VII. CONCLUSION

As the EDR data is used as important evidence for vehicle accidents, the EDR data must be protected against data forgery attacks. However, the existing data protection methods for the EDR data are not secure against data forgery attacks when the logging procedure is not properly finalized in emergency situations. In light of this, we propose TB-Logger dealing with data verification issues that may occur in emergency situations. To the best of our knowledge, TB-Logger is the first method that can provide data verification in emergency situations. TB-Logger provides MAC and digital signature functions to protect the EDR data via TEE. If an emergency causes an incomplete data block to be created without a corresponding digital signature, TB-Logger can still verify the incomplete data block using blockchain and Shamir's (t, n) secret sharing. Therefore, the data forgery attacks on the incomplete data block can be detected by TB-Logger. In order to evaluate the overhead of TB-Logger, an experiment was conducted using two real vehicles, Hyundai Avante CN7 and Tesla Model 3, and we confirmed that CAN packets were recorded in TB-Logger without data loss. In addition, through experiments simulating emergency situations, it was confirmed that there is an incomplete data block that can be handled by the TB-Logger but cannot be dealt with by the existing logging data protection methods.

REFERENCES

[1] European Commission. (2020). *Event Data Recorder*. Accessed: May 9, 2022. [Online]. Available: https://circabc.europa.eu

[2] *IEEE Standard for Motor Vehicle Event Data Recorders (MVEDRs) Amendment 1: MVEDR Connector Lockout Apparatus (MVEDRCLA)*, IEEE Standard 1616a, 2010, pp. 1–19.

[3] National Highway Traffic Safety Administration. (2006). *NHTSA 49 CFR Part 563 Final Regulatory Evaluation: Event Data Recorders (EDRs)*. Accessed: May 9, 2022. [Online]. Available: https://www.nhtsa.gov/sites/nhtsa.gov/files/fmvss/EDRFRIA.pdf

[4] National Highway Traffic Safety Administration. (2006). *NHTSA 49 CFR Part 563 Final rule: Data Recorders*. Accessed: May 9, 2022. [Online]. Available: https://www.nhtsa.gov

[5] National Highway Traffic Safety Administration. (2006). *NHTSA 49 CFR Part 563 Frequently Asked Questions and Additional Information about Event Data Recorders*. Accessed: May 9, 2022. [Online]. Available: https://www.nhtsa.gov

[6] Bellas & Wachowski Attorneys at Law. *The Admissibility of EDR Evidence in Civil and Criminal Cases*. Accessed: May 9, 2022. [Online]. Available: https://www.bellas-wachowski.com

[7] M. Kim and K.-Y. Kim, "Data forgery detection for vehicle black box," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2014, pp. 636–637.

[8] R. White, G. Caiazza, A. Cortesi, Y. I. Cho, and H. I. Christensen, "Black block recorder: Immutable black box logging for robots via blockchain," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3812–3819, Oct. 2019.

[9] S. Lee, W. Choi, H. J. Jo, and D. H. Lee, "T-box: A forensics-enabled trusted automotive data recording method," *IEEE Access*, vol. 7, pp. 49738–49755, 2019.

[10] S. Lee, H. J. Jo, W. Choi, H. Kim, J. H. Park, and D. H. Lee, "Fine-grained access control-enabled logging method on ARM trustzone," *IEEE Access*, vol. 8, pp. 81348–81364, 2020.

[11] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979, doi: 10.1145/359168.359176.

[12] M. P. DaSilva. (2008). *Analysis of Event Data Recorder Data for Vehicle Safety Improvement*. Accessed: May 9, 2022. [Online]. Available: https://rosap.ntl.bts.gov/view/dot/6182

[13] H. C. Gabler, C. E. Hampton, and J. Hinch. (2004). *Crash Severity: A Comparison of Event Data Recorder Measurements With Accident Reconstruction Estimates*. Accessed: May 9, 2022. [Online]. Available: https://www.sae.org/publications/technical-papers/content/2004-01-1194

[14] Y. Yao and E. Atkins, "The smart black box: A value-driven high-bandwidth automotive event data recorder," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1484–1496, Mar. 2021, doi: 10.1109/TITS.2020.2971385.

[15] Squarell Technology. (2017). *Squarell EDR*. Accessed: May 9, 2022. [Online]. Available: https://squarell.com

[16] Greg Coleman Law. *Vehicle Tracking Devices and How They Could Impact Your Claim*. Accessed: May 9, 2022. [Online]. Available: https://www.gregcolemanlaw.com

[17] W. P. Risk, G. S. Kino, and H. J. Shaw, "Fiber-optic frequency shifter using a surface acoustic wave incident at an oblique angle," *Opt. Lett.*, vol. 11, no. 2, pp. 115–117, Feb. 1986. [Online]. Available: http://opg.optica.org/ol/abstract.cfm?URI=ol-11-2-115

[18] ARM Developer. *ARM Security Technology Building a Secure System using TrustZone Technology*. Accessed: May 9, 2022. [Online]. Available: https://developer.arm.com

[19] M. Bellare and B. Yee, "Forward integrity for secure audit logs," Univ. California San Diego, San Diego, CA, USA, Tech. Rep., 1997.

[20] M. Bellare, "Forward-security in private-key cryptography," in *Proc. Cryptographers' Track RSA Conf.*, 2003, pp. 1–18.

[21] B. Schneier and J. Kelsey, "Cryptographic support for secure logs on untrusted machines," in *7th USENIX Secur. Symp. (USENIX Secur.)*, Jan. 1998, pp. 1–11.

[22] U. Chelladurai and S. Pandian, "A novel blockchain based electronic health record automation system for healthcare," *J. Ambient Intell. Humanized Comput.*, vol. 13, no. 1, pp. 693–703, Jan. 2022.

[23] A. Ali, A. Khan, M. Ahmed, and G. Jeon, "BCALS: Blockchain-based secure log management system for cloud computing," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 4, p. e4272, 2022.

[24] K. T. Akhter Md Hasib, I. Chowdhury, S. Sakib, M. Monirujjaman Khan, N. Alsufyani, A. Alsufyani, and S. Bourouis, "Electronic health record monitoring system and data security using blockchain technology," *Secur. Commun. Netw.*, vol. 2022, Feb. 2022, Art. no. 2366632.

[25] H. B. Mahajan, A. S. Rashid, A. A. Junnarkar, N. Uke, S. D. Deshpande, P. R. Futane, A. Alkhayyat, and B. Alhayani, "Integration of healthcare 4.0 and blockchain into secure cloud-based electronic health records systems," *Appl. Nanoscience*, vol. '13, pp. 2329–2342, Feb. 2022.

[26] A. Tchagna Kouanou, C. Tchito Tchapga, M. Sone Ekonde, V. Monthe, B. A. Mezatio, J. Manga, G. R. Simo, and Y. Muhozam, "Securing data in an Internet of Things network using blockchain technology: Smart home case," *Social Netw. Comput. Sci.*, vol. 3, no. 2, pp. 1–10, Mar. 2022.

[27] Y. Wang, Z. Su, N. Zhang, J. Chen, X. Sun, Z. Ye, and Z. Zhou, "SPDS: A secure and auditable private data sharing scheme for smart grid based on blockchain," *IEEE Trans. Ind. Informat.*, vol. 17, no. 11, pp. 7688–7699, Nov. 2021.

[28] N. B. Truong, K. Sun, G. M. Lee, and Y. Guo, "GDPR-compliant personal data management: A blockchain-based solution," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1746–1761, 2020.

[29] S. Lee, W. Choi, H. J. Jo, and D. H. Lee, "How to securely record logs based on ARM trustzone," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, New York, NY, USA, Jul. 2019, pp. 664–666, doi: 10.1145/3321705.3331001.

[30] D. Lu, M. Shi, X. Ma, X. Liu, R. Guo, T. Zheng, Y. Shen, X. Dong, and J. Ma, "Smaug: A TEE-assisted secured SQLite for embedded systems," *IEEE Trans. Dependable Secure Comput.*, pp. 1–18, 2022.
[31] The New York Times. (2013). *A Black Box for Car Crashes*. Accessed on Dec. 20, 2022. Accessed: Dec. 20, 2022. [Online]. Available: https://www.nytimes.com
[32] M. Lipp, D. Gruss, R. Spreitzer, C. Maurice, and S. Mangard, "ARMageddon: Cache attacks on mobile devices," in *Proc. 25th USENIX Secur. Symp. (USENIX Secur.)*. Austin, TX, USA, Aug. 2016, pp. 549–564.
[33] D. Cerdeira, N. Santos, P. Fonseca, and S. Pinto, "SoK: Understanding the prevailing security vulnerabilities in trustzone-assisted TEE systems," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2020, pp. 1416–1432.
[34] S. Wan, M. Sun, K. Sun, N. Zhang, and X. He, "RusTEE: Developing memory-safe ARM trustzone applications," in *Proc. Annu. Comput. Secur. Appl. Conf.*, Dec. 2020, pp. 442–453.
[35] N.-Y. Ahn and D. Hoon Lee, "Countermeasure against side-channel attack in shared memory of trustzone," 2017, *arXiv:1705.08279*.
[36] H. J. Jo and W. Choi, "BPRF: Blockchain-based privacy-preserving reputation framework for participatory sensing systems," *PLoS ONE*, vol. 14, no. 12, pp. 1–23, Dec. 2019, doi: 10.1371/journal.pone.0225688.
[37] K. Yasuda, "Multilane HMAC—Security beyond the birthday limit," in *Progress in Cryptology—(INDOCRYPT)*, K. Srinathan, C. P. Rangan, and M. Yung, Eds. Berlin, Germany: Springer, 2007, pp. 18–32.
[38] Linaro Limited. *OP-TEE*. Accessed: May 9, 2022. [Online]. Available: https://www.op-tee.org

DONGWOO KANG received the B.S. degree in computer engineering from Hallym University, Chuncheon, South Korea, in 2021. He is currently pursuing the M.S. degree with the Graduate School of Software, Soongsil University, Seoul, South Korea. His research interests include system security and secure logging.

HYO JIN JO received the B.S. degree in industrial engineering and the Ph.D. degree in information security from Korea University, Seoul, South Korea, in 2009 and 2016, respectively. He was a Postdoctoral Researcher with the Department of Computer and Information Systems, University of Pennsylvania, Philadelphia, PA, USA, from 2016 to 2018. He is currently an Assistant Professor with the School of Software, Soongsil University, Seoul. His research interests include applied cryptography and vehicle security.