IEEE Open Journal of Vehicular Technology

Received 12 August 2022; accepted 24 October 2022. Date of publication 27 October 2022;
date of current version 9 December 2022. The review of this article was coordinated by Editor Yongdong WU.

Digital Object Identifier 10.1109/OJVT.2022.3217552

The Security of "2FLIP" Authentication Scheme for VANETs: Attacks and Rectifications

MIR ALI REZAZADEH BAEE [1] (Senior Member, IEEE), LEONIE SIMPSON [1], ERNEST FOO [2,3] (Member, IEEE), AND JOSEF PIEPRZYK [4,5,6]

[1]School of Computer Science, Queensland University of Technology, Brisbane, QLD 4000, Australia
[2]School of Information and Communication Technology, Griffith University, Brisbane, QLD 4111, Australia
[3]School of Computer Science, Queensland University of Technology, Brisbane, QLD 4000, Australia
[4]Commonwealth Scientific and Industrial Research Organization, Data61, Marsfield, NSW 2122, Australia
[5]Institute of Computer Science, Polish Academy of Sciences, 01-248 Warsaw, Poland
[6]School of Computer Science, Queensland University of Technology, Brisbane, QLD 4000, Australia

CORRESPONDING AUTHOR: MIR ALI REZAZADEH BAEE (e-mail: baee@ieee.org)

The work of Mir Ali Rezazadeh Baee was supported by the Queensland University of Technology Postgraduate Research Award. The work of Josef Pieprzyk was supported in part by the Australian Research Council under Grant DP180102199 and in part by the Polish National Science Center (Narodowe Centrum Nauki) under Grant 2018/31/B/ST6/03003.

ABSTRACT Wireless broadcast transmission enables Inter-vehicle or Vehicle-to-Vehicle (V2V) communication among nearby vehicles and with nearby fixed equipment, referred to as Road Side Units (RSUs). The vehicles and RSUs within transmission range establish a self-organizing network called Vehicular Ad-hoc Network (VANET). The V2V communication in VANETs is vulnerable to cyber-attacks involving message manipulation. Thus, mechanisms should be applied to ensure both the authenticity and integrity of the data broadcast. However, due to privacy concerns, it is important to avoid the use of identifiers that may aid tracking and surveillance of drivers. This is a serious constraint on authentication mechanisms. Recently, Wang et al. [1] proposed *A Two-Factor Lightweight Privacy Preserving Authentication Scheme for VANET* named 2FLIP. They claim that their scheme includes a secure systemkey update protocol to restore the whole system when necessary. In this paper, we show that this is incorrect: 2FLIP does not provide perfect forward secrecy. This results in a known-key attack, as well as message forgery attack by an external adversary who may be an unregistered vehicle user. This external adversary can generate valid anonymous messages and further, they cannot be traced. The 2FLIP scheme is efficient, so we propose a modification to improve the security. We provide a formal security proof to show that our proposal is indeed provably secure. We demonstrate the efficiency of our proposal by conducting extensive performance analysis. We believe the enhanced system-key update protocol will be useful for application by researchers and designers in current and future VANET authentication schemes.

INDEX TERMS Authentication, cryptography, known-key attack, message forgery attack, perfect forward secrecy.

I. INTRODUCTION

Cooperative Intelligent Transport System (C-ITS) is an emerging technology with the potential to improve road safety [2]. Car manufacturers embed devices such as IEEE 802.11p in vehicles to enable wireless communication with other vehicles and with nearby fixed equipment, referred to as Road Side Units (RSUs). This enables devices (vehicles and RSUs) within transmission range to establish a self-organizing network called Vehicular Ad-hoc Network (VANET) [3]. The VANET exchanges beacon messages at a high update rate carrying critical information [4], [5].

A. NETWORK MODEL AND ASSUMPTIONS

The vehicular network model consists of a Certificate Authority (CA), RSUs along the roads, On-Board Units (OBUs) embedded in vehicles, and communication between these entities. This VANET architecture and communication model is illustrated in Fig. 1.

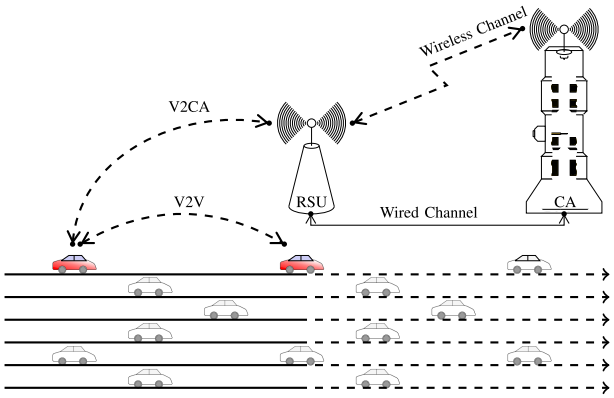

FIGURE 1. Vehicular network model.

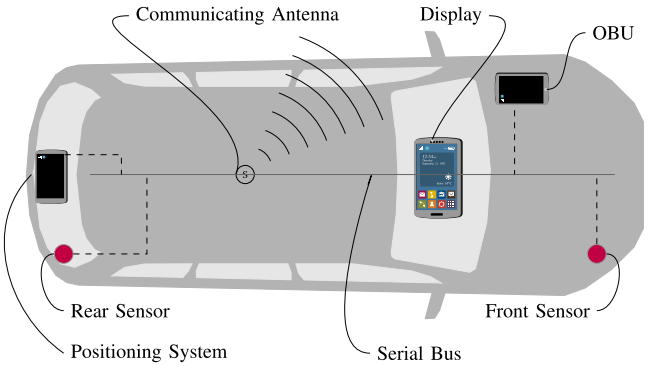

FIGURE 2. Envisioned smart vehicle prototype.

The CA is assumed to be always online, secure, and fully trusted by other network entities. The CA can be implemented in a multi-layer structure; for example, with a root CA and several sub-CAs. For sake of simplicity, here we show the CA as a single entity. The CA is a managing authority that can act as the root of trust to generate, update, and revoke credentials for other network entities, including vehicles. For example, the department of motor vehicles or the Department of Transportation (DOT) can act as the root CA.

The distributed RSUs are equipped with a higher computational capability and transmission power than OBUs. As shown in Fig. 1, the CA and RSUs can connect to each other through wired (solid line) and/or wireless (half-dashed line) channels. The RSUs work as gateways to deliver data from the CA to roadside vehicles, and vice versa. The range of an RSU-to-vehicle communication can be larger than that of the V2V and vehicle-to-RSU communications [6].

Smart vehicles equipped with OBU, sensors, and Global Positioning System (GPS) move along the roads, and communicate with other vehicles and RSUs according to a defined Intelligent Transportation Systems Radio Service (ITS-RS) standard, such as the Dedicated Short Range Communications (DSRC) protocol [6]. A Tamper Proof Device (TPD) is embedded in each OBU to store the user inaccessible cryptographic keying materials involved in cryptographic operations. Moreover, each vehicle has a unique barcode/identifier that is known to the DOT. Fig. 2 illustrates an envisioned smart vehicle prototype.

This system is useful if all messages are legitimate. Safety messages are broadcast to reach all vehicles within communication range. However, malicious entities could manipulate messages. Mechanisms should be applied to ensure both identification of the data source (entity authentication) and authentication of the message (assurance of data origin and data integrity) [7].

Authentication requirements can often conflict with privacy requirements. A unique identifier, such as a Vehicle Identification Number (VIN), is provided to a vehicle for authentication purposes. However, this vehicle identifier can be associated with an identifiable individual (e.g., driver or vehicle owner). In this case, the exchanged data reveals personal information, (e.g., aggregating location information over time can reveal work or home addresses and travel patterns). An adversary can capture communications and link the identifiers to specific vehicles, and consequently to the drivers (ID disclosure), providing a means for surveillance [8]. Hence, protection and confidentiality of data exchanged is required to avoid privacy breaches. For both identification and message authentication, protection of the driver's identity during authentication must be guaranteed.

B. EXISTING WORK AND RESEARCH CHALLENGE

Any cryptographic technique for authentication requires the use of a cryptographic key. A key-update mechanism is needed to refresh the key to ensure security, for example after a cryptographic-key breach occurs, or if a cryptographic-key is unknowingly accessed. The key-update mechanism must be carefully designed to satisfy both security against known-key attacks [9, §12.2.3], and perfect forward secrecy; if a cryptographic secret key for the current session is revealed, an adversary must not be able to use this information to assist in recovering messages from past and/or future ciphertexts [9, §12.16]. To achieve this, there must not be any exploitable dependencies among keys for different sessions (keying material).

Multiple authentication schemes have been proposed to secure V2V communication [10], [11] (for examples see [1], [5], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23]). Many of these authentication schemes (e.g., [12], [13], [14], [17], [18]) do not include a key-update protocol for long-term cryptographic keying material. This is a serious omission. Some authentication schemes include a key-update protocol. However, these may not have been carefully analyzed to ensure security against known-key attacks and perfect forward secrecy.

Wang et al. [1] propose *2FLIP: A Two-Factor Lightweight Privacy Preserving Authentication Scheme for VANET*. In 2FLIP, the TPD embedded in each vehicle is equipped with

a system key to generate Message Authentication Codes (MACs) for the broadcast messages. Recipient TPDs can verify the messages by creating MACs of the received messages with the same system key. This only requires one hash computation and one MAC operation to accomplish the message verification. Hence, the protocol is very efficient in terms of computation.

In 2FLIP, the same copy of system key must be stored in all vehicle TPDs. However, Huang et al. [17] view this as a single point of failure. If any single vehicle's system-key is compromised, all vehicles in the network will be affected (as they all have the same key), and all will need to be updated with a new system-key. Clearly, a key-update mechanism is required.

The authors of 2FLIP acknowledge the possibility of key compromise. They propose renewing the system key, and claim that the mechanism they provide can restore the system quickly in the event that the system key is leaked. They recognize the importance of the key-update mechanism and state that such a protocol is necessary for a complete secure system (see [1, §III – C]).

A serious security flaw in the system-key update mechanism of 2FLIP is reported by Baee et al. [24], [25]. There is a clear dependency between successive system keys. In the 2FLIP key-update protocol, a CA that is fully trusted by other network entities encrypts a new system key, treating the system key as the message content and encrypting this using the previous system key. In the case where a past system key is compromised, clearly the new key is not protected and very easily discovered.

As 2FLIP is computationally efficient and has some practical advantage, it would be a shame to discard it due to this system-key update flaw. This research reviews the scheme and suggests modifications to mitigate this flaw, to address the security of the 2FLIP scheme.

C. RESEARCH CONTRIBUTION

This paper contains five major contributions. First, the message signing and verifying protocol, system-key update protocol, and adversary model in 2FLIP scheme are reviewed, and a vulnerability in the system-key update protocol of 2FLIP scheme is identified. Secondly, two attacks that exploit this vulnerability are described and demonstrated. Thirdly, a modification to mitigate the security flaw in 2FLIP system-key update protocol is proposed. Fourthly, a formal security proof for the proposed system-key update protocol is given. This demonstrates that the new protocol is indeed provably secure. Finally, the efficiency of our protocol is demonstrated by conducting extensive performance analysis.

D. ORGANIZATION OF THE PAPER

The remainder of this study is organized as follows. Section II briefly reviews the message signing and verifying protocol, system-key update protocol, and adversary model in 2FLIP scheme. Section III demonstrates the vulnerabilities in 2FLIP.

TABLE 1. List of Abbreviations

Abbreviation	Meaning
ACT	Average Computation Time
AES-CTR	Advanced Encryption Standard in Counter Mode
ARIB	Association of Radio Industries and Businesses
BN	Barreto-Naehrig
C-ITS	Cooperative Intelligent Transportation System
CA	Certificate Authority
DOT	Department of Transportation
DSRC	Dedicated Short Range Communications
ECC	Elliptic Curve Cryptography
ECIES	Elliptic Curve Integrated Encryption Scheme
FIPS	Federal Information Processing Standard
GPS	Global Positioning System
HMAC	Hash-based Message Authentication Code
HKDF	Hash-based Key-Derivation Function
IND-CCA	Indistinguishability under Chosen-Ciphertext Attack
IND-CPA	Indistinguishability under Chosen-Plaintext Attack
ITS-RS	Intelligent Transportation Systems Radio Service
MAC	Message Authentication Code
NIST	National Institute for Standards and Technology
OBU	On-Board Unit
PID	Personal Identifier
PIN	Personal Identification Number
PPT	Probabilistic Polynomial-Time
PRF	PseudoRandom Function
RFC	Request For Comments
RSU	Road-Side Unit
SHA	Secure Hash Algorithm
SMS	Short Message Service
SUF-CMA	Strong Unforgeability under Chosen-Message Attack
TPD	Tamper Proof Device
V2CA	Vehicle-to-Certificate Authority
V2V	Vehicle-to-Vehicle
VANET	Vehicular Ad-hoc Network
VIN	Vehicle Identification Number

Section IV presents the modification proposed by this research, and Section V provides formal security proof to show the modification is indeed provably secure and preserves the privacy requirements. Section VI evaluates the computational cost and communication overhead of the proposed key-update protocol. Section VII discusses the results, and finally, the paper is summarized in Section VIII. For the convenience of the reader, a list of abbreviations and symbols used throughout the paper is given in Tables 1 and 2, respectively.

II. THE 2FLIP SCHEME

This section reviews the message signing/verifying protocol, system-key update protocol, and adversary model proposed in 2FLIP.

In addition to the notations presented in this section, everywhere in this paper the concatenation symbol "‖" denotes an operation that combines several strings into one string where the constituent strings are uniquely recoverable from the final one.

TABLE 2. List of Symbols

Symbol	Definition
$\|$	concatenation symbol
t_s	time at generating a message
t_r	time at receiving a message
Δt	predefined time threshold
$\mathrm{X} \rightarrow \mathrm{Y}$	a message transmission from X to Y
$\mathrm{X} \downarrow$	processing a message by X
$\mathrm{X} \rightleftharpoons \mathrm{Y}$	a two-way message transmission between X and Y
k_m	system key
σ_i	a MAC generated on message m by TPD_i
Enc	message encryption procedure
sg	an identity-based message signature
ID_{CA}	the CA's identifier
$\mathcal{A}$	Probabilistic Polynomial-Time (PPT) adversary
q	the order of a cyclic additive elliptic curve group $\mathbb{G}$
m	an arbitrary message
h	a simple hash function
(Enc, Dec)	a symmetric encryption/decryption scheme
${}_kC(m)$	encrypted data produced on message m using key k
$HMAC()$	a HMAC function
${}_k\delta(m)$	an output HMAC tag on message m using key k
p	a large prime number
$E(\mathbb{F}_q)$	an elliptic curve
$\mathbb{F}$	finite field
P	a random point of order q
$\langle P \rangle$	the cyclic subgroup of points generated by P
csc	a list of cipher suite components
PWD	a unique random password for the vehicle user
$salt_u$	a unique random value
CA_{par}	a set of system parameters
d_t	a temporary private key
Q_t	a temporary public key
ref	a MAC tag as reference number
Γ_{2FLIP}	the 2FLIP scheme
$\mathcal{F}^{\mathrm{skup}}$	our proposed system-key update protocol
$Adv_{\mathcal{A}}$	the probability taken over all coin tosses made by $\mathcal{A}$
λ	the security parameter
$\mathcal{S}$	a simulator for our protocol $\mathcal{F}^{\mathrm{skup}}$
G	a game played between $\mathcal{A}$ and $\mathcal{S}$
T_{mul}^{scal}	the ACT for the ECC base point scalar multiplication
T_{enc}^{aes}	the ACT for the 128-bit AES encryption in CTR mode
T_{dec}^{aes}	the ACT for the 128-bit AES decryption in CTR mode
T_{sha}^{hkdf}	the ACT for the HKDF with 256-bit SHA
T_{sha}^{hmac}	the ACT for the HMAC with 256-bit SHA

The symbols "t_s" denotes time at generating a message. The symbol "t_r" denotes time at receiving a message. A message receiver drops the message if the time interval between the time of receiving t_r and the time t_s exceeds a predefined threshold denoted by "Δt". This is used to determine the freshness of the message.

Assume X and Y are two communicating nodes. The symbol "X $\rightarrow$ Y" denotes a message transmission from X to Y. The symbol "X $\downarrow$" denotes processing a message by X, before transmitting a message or after receiving a message. The symbol "X $\rightleftharpoons$ Y" denotes a two-way message transmission from X to Y, and also from Y to X.

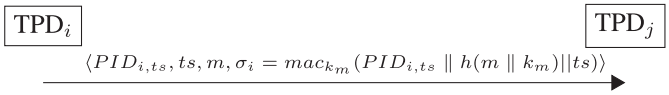

FIGURE 3. Message broadcasting in 2FLIP.

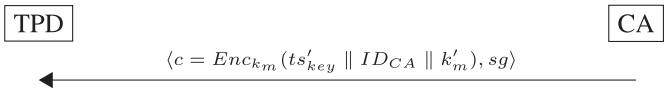

FIGURE 4. System-key update protocol in 2FLIP.

A. MESSAGE SIGNING AND VERIFYING IN 2FLIP

Fig. 3 shows the message exchange in this phase. The same copy of system key k_m is stored in all TPDs. The following steps are executed by two vehicle TPDs i and j in this phase:

1) A TPD_i generates a MAC $\sigma_i = mac_{k_m}(PID_{i,ts} \parallel h(m \parallel k_m)||ts)$ on a message m using k_m, where PID_i is a vehicle's Personal Identifier (PID) preloaded by the CA on the vehicle TPD (during system initialization and entity registration phase [1, §IV – A]), h is a simple hash function, and ts is current time (see [1, §IV – C, and D]).
2) TPD_i broadcasts the tuple $\{PID_{i,ts}, \sigma_i, ts, m\}$.
3) A receiver TPD_j calculates $\sigma_i^* = mac_{km}(PID_{i,ts} \parallel h(m \parallel k_m) \parallel ts)$ to verify the legitimacy of the received broadcast tuple $\{PID_{i,ts}, \sigma_i, ts, m\}$, and accepts the message if $\sigma_i^* = \sigma_i$; otherwise, rejects the message.

B. SYSTEM-KEY UPDATE IN 2FLIP

Fig. 4 shows the message exchange in this phase. In 2FLIP, the CA is responsible for updating vehicle TPDs with new system key. The following steps are executed by vehicle TPDs and CA in this phase:

TPD $\leftarrow$ CA

1) The CA with identifier ID_{CA} encrypts a new system key k'_m under the previous system key k_m such that $c = Enc_{k_m}(ts'_{key} \parallel ID_{CA} \parallel k'_m)$. Note that Enc is an encryption procedure, and ts'_{key} is time at generating this new system key.
2) The CA broadcasts the message $\{c, sg\}$ to vehicles in the network through the RSUs. Note that sg is the output of an identity-based message signing function (see [1, §IV – E]).

TPD $\downarrow$

1) The TPD receives the key-update message $\{c, sg\}$, decrypts c, and recovers the payload $\{ts'_{key} \parallel ID_{CA} \parallel k'_m\}$. Then, the TPD checks the ts'_{key} to prevent the replay attack. After that, the TPD verifies ID_{CA} with the previously published ID_{CA} to authenticate the origin of this message. Finally, verifies the signature sg to ensure that the message is valid.
2) The TPD updates k_m to k'_m and ts_{key} to ts'_{key}.

C. ADVERSARY MODEL IN 2FLIP

Let $\mathcal{A}$ be a Probabilistic Polynomial-Time (PPT) adversary. In the computational world, the PPT adversary $\mathcal{A}$ is modeled as a PPT Turing machine [26]. Wang et al. [1] assumed that $\mathcal{A}$ has impressive communication abilities through powerful receivers, can control the communication channel, and monitor on-the-fly data exchange, as well as tamper with messages and replace the original messages with modified messages. They assume that the cryptographic keying materials are kept safe in TPDs and key disclosure would never be achieved (see [1, §III – B]).

III. ATTACKS ON 2FLIP SCHEME

In 2FLIP, a key-update procedure is performed if a cryptographic key breach occur, or if a cryptographic key is unknowingly accessed (see [1, §III – C]). However, the procedure uses an old system key to encrypt a new system key (key wrapping). This is absolutely a vulnerability. It produces dependencies among keying material (see [1, §IV – E], or refer to Section II-B). In the case where a past system key k_m is compromised, the new key k'_m is very easily compromised.

A. KNOWN-KEY ATTACK

In 2FLIP, a known system key k_m is used by all vehicle TPDs. If a single vehicle's system key is compromised, all vehicles in the network will need to be updated with a new system key. However, the key-update phase proposed in 2FLIP is not secure. Clearly, it is a simple matter for the adversary to successfully perform a known-key attack:

1) The adversary is aware of a compromised key k_m.
2) The adversary eavesdrops $c = Enc_{k_m}(ts'_{key} \parallel ID_{CA} \parallel k'_m)$ on the broadcast message $\{c, sg\}$ (refer to Section II-B).
3) The adversary decrypts $c = Enc_{k_m}(ts'_{key} \parallel ID_{CA} \parallel k'_m)$ using the known system key k_m, and recovers the new system key k'_m.

B. MESSAGE FORGERY ATTACK

Lack of perfect forward secrecy results in message forgery attack. Wang et al. [1] claim that 2FLIP is resistant to forgery or modification of message because the adversary cannot forge or modify the message $\{PID_{i,ts}, \sigma_i, ts, m\}$ (refer to Section II-A) to let it be accepted by other vehicles (see [1, §V – B]).

This research proves that an attacker (an external adversary who is an unregistered vehicle user) can launch this attack efficiently. The following steps can be executed by the adversary to successfully perform a message forgery attack:

1) The adversary $\mathcal{A}$ randomly picks a personal identifier $PID_i \in Z_q^*$, where q is order of a cyclic additive group $\mathbb{G}$.
2) The adversary uses the k_m compromised in the key-update phase (explained in Section III-A) to generate a valid tuple $\{PID_{\mathcal{A},ts}, \sigma_{\mathcal{A}}, ts, m\}$ and broadcast for a long period of time, where $\sigma_{\mathcal{A}} = mac_{k_m}(PID_{i,ts} \parallel h(m \parallel k_m)||ts)$, m is an arbitrary message, h is a simple hash function, and ts is current time.
3) A receiver TPD$_i$ calculates $\sigma^*_{\mathcal{A}} = mac_{km}(PID_{\mathcal{A},ts} \parallel h(m \parallel k_m) \parallel ts)$ to verify the legitimacy of the received broadcast tuple $\{PID_{\mathcal{A},ts}, \sigma_{\mathcal{A}}, ts, m\}$, and accepts the message as $\sigma^*_{\mathcal{A}} = \sigma_{\mathcal{A}}$.

IV. THE PROPOSED COUNTERMEASURE

This section proposes a modification to improve the 2FLIP system-key update protocol. Since the scheme is efficient, this research slightly modifies the protocol to improve its security (Section IV-D). Seven steps are added to the *System Initialization and Entity Registration* phase of 2FLIP (Section IV-C), and propose an alternative *System-key Update* protocol.

A. SECURITY OBJECTIVES AND DESIGN GOALS

This section focuses on security objectives of the proposed system-key update protocol, including: message integrity and authentication, confidentiality of the new system key, anonymity and unlinkability, and perfect forward secrecy.

Message Integrity and Authentication: Message authentication provides assurance of data origin, and also data integrity [9]. The authentication process must be secure against an adversary trying to fabricate (forge) an authentication tag for a message without having access to the respective legitimate sender's secret credential.

Confidentiality of New System key: This is to ensures that no efficient adversary can infer any information about the new system key sent to the vehicle TPD.

Anonymity and Unlinkability: It should be impossible for an adversary and the network entities (except the CA) to link a message to a vehicle/driver correctly. If an observer tried to guess which vehicle transmitted a particular message, there should be only a low probability of linking a vehicle's actions or identifying it among the set of all vehicles.

Perfect Forward Secrecy: Ensures that compromise of past session keys does not allow an adversary to compromise future session keys [9, §12.17]. A protocol is vulnerable to a known-key attack if perfect forward secrecy is not provided.

B. BUILDING BLOCKS

The modification proposed in this research involves the use of Elliptic Curve Integrated Encryption Scheme (ECIES) proposed by Abdalla et al. [27]. The ECIES is a hybrid public-key encryption scheme, and employs the Diffie-Hellman protocol [28] over elliptic curves to establish a symmetric encryption key. This research uses ECIES with the (well-known) cryptographic primitives. This section briefly reviews these cryptographic primitives and their underlying security assumptions.

1) SYMMETRIC ENCRYPTION SCHEME

Let (Enc, Dec) denote a symmetric encryption scheme which provides Indistinguishability under Chosen-Plaintext Attacks (IND-CPA): given two messages of equal length, a challenger randomly decides to encrypt one of the messages and to

the adversary. It should be hard for the adversary to distinguish which of the messages was encrypted. The symmetric encryption operation $Enc(m, k_1)$ outputs ciphertext ${}_{k_1}C(m)$: a plaintext m is encrypted under symmetric key k_1. The corresponding decryption operation $Dec({}_{k_1}C(m), k_1)$ outputs message m: ciphertext ${}_{k_1}C(m)$ is decrypted under symmetric key k_1. The maximum advantage of PPT adversary $\mathcal{A}$ in breaking the IND-CPA property of (Enc, Dec) is denoted by $Adv^{\text{ind-cpa}}_{\mathcal{A}(Enc,Dec)}$. This proposal uses the Advanced Encryption Standard in Counter Mode (AES-CTR) for symmetric encryption, as specified in National Institute for Standards and Technology (NIST) Special Publication (SP) 800-38A [29].

2) HASH-BASED MESSAGE AUTHENTICATION CODE

Let $HMAC()$ denote a Hash-based Message Authentication Code (HMAC) function which is believed to satisfy Strong Unforgeability under Chosen-Message Attacks (SUF-CMA): it should be computationally infeasible for the adversary to find a valid pair of message and tag whether the message is new and has never been signed by a legitimate signer, or the message is old and the new tag is not previously attached to this message by a legitimate signer [30]. The HMAC operation $HMAC(m, k_2)$ outputs tag ${}_{k_2}\delta(m)$: a tag is generated on message m under symmetric key k_2. To verify a received message-tag pair, a new HMAC tag ${}_{k_2}\delta'(m)$ is generated on the message and the tags are compared: the tag ${}_{k_2}\delta(m)$ is accepted if, and only if, ${}_{k_2}\delta'(m) = {}_{k_2}\delta(m)$. The maximum success probability of PPT adversary $\mathcal{A}$ in finding a forgery is denoted by $Succ^{\text{suf-cma}}_{\mathcal{A}(HMAC)}$, where $\mathcal{A}$ is given access to the tagging/verification oracle. This proposal uses the HMAC for message authentication, as specified in the (U.S.) *Federal Information Processing Standard* (FIPS) 198-1 [31], as well a Secure Hash Algorithm $Hash$, as specified in the FIPS 180-4 [32].

3) HASH-BASED KEY-DERIVATION FUNCTION

Let $HKDF()$ denote a Hash-based Key-Derivation Function (HKDF) based on the HMAC presented in Section IV-B2 which extracts an input master secret key k, and expands it into several additional pseudorandom secret keys [33]. For instance, two secret keys k_1 and k_2 can be derived (extract process) such that $k_1 = HKDF(1, k)$ and $k_2 = HKDF(2, k)$, where 1 and 2 are salt values. The HKDF is defined to operate with and without random salt [34]. The security of HKDF as a HMAC-based KDF is based on the assumption that the compression function of the underlying hash is itself a PseudoRandom Function (PRF) [35]. The maximum advantage of PPT adversary $\mathcal{A}$ in distinguishing the outputs of PRF better than by a random guess and breaking security of the PRF is denoted by $Adv^{\text{ind-sec}}_{\mathcal{A}(PRF)}$. This proposal uses the HKDF to convert a shared-secret into key material suitable for use in encryption, integrity checking or authentication, as specified in the Request For Comments (RFC) 2898 [36].

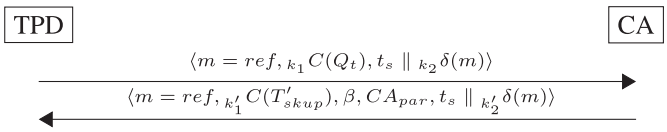

FIGURE 5. The new system-key update protocol.

C. SYSTEM INITIALIZATION AND ENTITY REGISTRATION

To address the perfect forward secrecy issue, this section adds seven steps to the original version of the protocol in the *System Initialization and Entity Registration* phase. The CA generates system parameters and cipher suite components. The following steps are executed by the CA and a vehicle user in this phase:

1) The CA chooses a large prime p, a random point P of order q, and an elliptic curve $E(\mathbb{F}_q)$ over finite field $\mathbb{F}$ to satisfy the *Short Weierstrass form* of equation, $y^2 = x^3 + ax + b \pmod q$ [37], where $q = p^m$ for $m \geq 1$ is the smallest positive integer such that $q.P = \mathcal{O}$, $a, b \in \mathbb{F}_q$ are the constant coefficients of the curve, and $\langle P\rangle$ be the cyclic subgroup of points generated by P.
2) The CA generates system parameters $CA_{par} = \{p, q, a, b, P, csc\}$, where csc is a list of cipher suite components applicable in the network (e.g., NIST P-256 curve and SHA-256).
3) The CA computes password $PWD_u = HKDF(salt_u, pwd_u)$ for the vehicle user, where $salt_u$ and pwd_u are two long unique values (e.g., 16 bytes each) generated truly at random.
4) The CA stores the hash value PWD_u in its database.
5) The CA stores $salt_u$ in the vehicle TPD.
6) The CA delivers pwd_u to the user securely.
7) The user provides further identity information (e.g., national identity/security number), a valid mobile phone number, or alternatively requests a hand-held password generator token. The token generates a password which is synchronized with the server, and is valid for a short period of time (e.g., one minute).

D. AN ALTERNATIVE SYSTEM-KEY UPDATE PROTOCOL

This section proposes an alternative system-key update protocol. Fig. 5 shows the message exchange in this phase. The modification is based on the use of vehicle user's mobile phone or a password generator token. The CA can update all TPD related keying materials and the user's salt value $salt_u$ without relying on the previous keys used by the CA and the TPDs. In addition, the CA can generate a new set of system parameters CA_{par} including its new public/private key pair, and update the vehicles with the new parameters. The key-update procedure can be called by a user, or advertised by the CA through RSUs; for example, at regular intervals (e.g., every 6 months) or when a long-term key is leaked. The following steps are executed by the vehicle and the CA in this phase:

If the user does not have a token, the following steps need to be executed by the user and the CA in this phase:

User $\rightleftharpoons$ CA

1) The user sends a key-update request to the CA through a Short Message Service (SMS).
2) The CA locates the PID_i corresponding to the user's mobile phone number in its database.
3) The CA generates a Personal Identification Number (PIN) pin_{phone} for the user.
4) The CA computes a symmetric key $k_{up} = HKDF(pin_{phone}, PWD_u)$, where $PWD_u = HKDF(salt_u, pwd_u)$ is the user's password saved in the CA's database (refer to Section IV-C).
5) The CA derives a secret key k_3 such that $k_3 = HKDF(3, k_{up})$.
6) The CA generates a HMAC tag $ref = {}_{k_3}\delta(0)$ for use as a reference number.
7) The CA sends pin_{phone} to the user's mobile phone through SMS.
8) The CA links ref and k_{up} to PID_i in its database for a short period of time (e.g., 1 minute).

Those users who are provided a hand-held password generator token by the CA can present the PIN pin_{token} to proceed this phase. This PIN is a password generated by the token and is synchronized with the CA. The PIN is valid for a short period of time (e.g., 1 minute). If the user has a token, the following steps need to be executed by the CA periodically (similar to the token's password changing rate) in this phase:

CA $\downarrow$

1) The CA computes a symmetric key $k_{up} = HKDF(pin_{token}, PWD_u)$.
2) The CA derives a secret key k_3 such that $k_3 = HKDF(3, k_{up})$.
3) The CA generates a HMAC tag $ref = {}_{k_3}\delta(0)$ as a reference number.
4) The CA links ref and k_{up} to PID_i in its database and always replaces a new generated ref value with the old one and updates the link.

The following steps are executed by the vehicle TPD and the CA in this phase:

TPD $\rightarrow$ CA

1) The TPD asks the user to enter the pwd_u through the OBU's computing interface. The pwd_u is the user's registration password which is created in the entity registration phase (refer to Section IV-C).
2) The TPD loads $salt_u$ and computes $PWD_u = HKDF(salt_u, pwd_u)$.
3) The TPD asks the user to enter the pin_{phone} through the OBU's computing interface and computes a key $k_{up} = HKDF(pin_{phone}, PWD_u)$, if this is a phone-based key-update.
4) The TPD asks the user to enter the pin_{token} through the OBU's computing interface and computes a key $k_{up} = HKDF(pin_{token}, PWD_u)$, if this is a token-based key-update.
5) The TPD derives three secret keys k_1, k_2, and k_3 such that $k_1 = HKDF(1, k_{up})$, $k_2 = HKDF(2, k_{up})$, and $k_3 = HKDF(3, k_{up})$.
6) The TPD generates a HMAC tag $ref = {}_{k_3}\delta(0)$ as a reference number (similar to the CA's generated reference number).
7) The TPD chooses a random temporary private key $d_t \in {}_R\mathbb{Z}_q^*$ and computes a temporary public key $Q_t = d_t.P$.
8) The TPD computes ciphertext ${}_{k_1}C(Q_t)$. Note that the key Q_t must be encrypted, because it is used to establish a Diffie-Hellman shared secret key by the CA when it responds to this request. Otherwise, the adversary can forge an authentication tag on any malicious response.
9) The TPD generates a HMAC tag ${}_{k_2}\delta(m)$ on message $m = \{ref, {}_{k_1}C(Q_t), t_s\}$.
10) The TPD sends tuple $\{m \parallel {}_{k_2}\delta(m)\}$ to the CA. This tuple is only sent once, which does not harm the privacy requirements (anonymity and unlinkability).

TPD $\leftarrow$ CA

1) The CA determine the freshness, and accepts the message if, $0 \leq t_r - t_s \leq \Delta t$, otherwise drops the message.
2) The CA reads reference number ref in message m, and searches for the corresponding PID_i linked to ref in its database (the CA drops the message if nothing is found).
3) The CA verifies that the vehicle PID_i is not recently blocked because of malicious activity, otherwise stops the process.
4) The CA loads the corresponding k_{up} value linked to PID_i and derives two secret keys k_1 and k_2 such that $k_1 = HKDF(1, k_{up})$ and $k_2 = HKDF(2, k_{up})$.
5) The CA generates a new HMAC tag ${}_{k_2}\delta'(m)$ and accepts the message if, and only if, ${}_{k_2}\delta'(m) = {}_{k_2}\delta(m)$. This is to ensure that message m has not been altered.
6) The CA decrypts ciphertext ${}_{k_1}C(Q_t)$ and recovers the vehicle TPD's temporary public key Q_t.
7) The CA chooses a random integer $b \in {}_R\mathbb{Z}_q^*$, and computes $\beta = b.P$ and $k' = b.Q_t$.
8) The CA derives two secret keys k_1' and k_2' such that $k_1' = HKDF(1, k')$ and $k_2' = HKDF(2, k')$.
9) The CA chooses a new system key k_m' as specified in the original *System Initialization and Entity Registration* protocol proposed in 2FLIP [1, §IV – A].
10) The CA generates a response $T'_{skup} = \{k_m', salt_u\}$ and computes ciphertext ${}_{k_1'}C(T'_{skup})$.
11) The CA generates a HMAC tag ${}_{k_2'}\delta(m)$ on message $m = \{ref, {}_{k_1'}C(T'_{skup}), \beta, CA_{par}, t_s\}$.
12) The CA sends tuple $\{m \parallel {}_{k_2'}\delta(m)\}$ to the TPD.
13) The CA removes k_{up} and ref under the vehicle PID_i in its database.

TPD $\downarrow$

1) The TPD corresponding to ref determines the freshness, and accepts the message if, $0 \leq t_r - t_s \leq \Delta t$, otherwise drops the message.

2) The TPD computes $k' = d_t.\beta$. It works because $d_t.\beta = d_t.(b.P) = b.(d_t.P) = b.Q_t$.
3) The TPD derives two secret keys k'_1 and k'_2 such that $k'_1 = HKDF(1, k)$ and $k'_2 = HKDF(2, k')$.
4) The TPD generates a new HMAC ${}_{k'_2}\delta'(m)$ and accepts the message if, and only if, ${}_{k'_2}\delta'(m) = {}_{k'_2}\delta(m)$, otherwise drops the message. This is to ensure that message m has not been altered.
5) The TPD decrypts ciphertext ${}_{k'_1}C(T'_{skup})$ and extracts $T'_{skup} = \{k'_m, salt_u\}$.
6) The TPD stores the new system key k'_m, $salt_u$, and CA_{par}.

V. SECURITY AND PRIVACY ANALYSIS

This section defines a security model, and conducts a security proof to show that our proposed system-key update protocol $\mathcal{F}^{\texttt{skup}}$ for 2FLIP scheme $\Gamma_{\texttt{2FLIP}}$ is indeed provably secure. Next, based on the requirements explained in the security objectives and design goals in Section IV-A, the security and privacy analysis are carried out in detail.

A. SECURITY MODEL AND DEFINITIONS

The actual security proof consists of a reduction of the underlying computational problem to an attack against the cryptographic scheme: if there exists an adversary that has a significant advantage against $\mathcal{F}^{\texttt{skup}}$, then there is an adversary to solve the computational problem. In the computational world the PPT adversary $\mathcal{A}$ is modeled as a PPT Turing machine [26]. This section shows that the view of $\mathcal{A}$ in the reduction remains unchanged from the one it has during a real attack.

Definition 1 (Adversary's Advantage): $\mathcal{A}$ can guess b' for a bit b where b is a fair coin. The advantage $Adv_{\mathcal{A}}$ is the probability taken over all coin tosses made by $\mathcal{A}$, which is:

$$Adv_{\mathcal{A}} = 2|Pr[b = b'] - 1/2|.$$

The advantage $Adv_{\mathcal{A}}$ measures that how much $\mathcal{A}$ is better than an adversary who simply guesses at random. An adversary who guesses at random has a success probability of $1/2$, and an advantage of $Adv = 0$. When the adversary always correctly finds the value of b', we have $Pr[b = b'] = 1$ and thus an advantage of $Adv = 1$.

Definition 2 (Negligible Function): A function $\epsilon : \mathbb{N} \rightarrow \mathbb{R}$ is negligible if for every positive polynomial $p(.)$ there exists an $N \in \mathbb{N}$ such that for all $N < n \in \mathbb{N}$ it holds that $\epsilon(n) < 1/p(n)$ [38].

Definition 3 (Security Model): Based on the network model and the adversary's ability, we define the security model of our proposed system-key update protocol $\mathcal{F}^{\texttt{skup}}$ for 2FLIP scheme $\Gamma_{\texttt{2FLIP}}$ through a game played between the PPT adversary $\mathcal{A}$ and a simulator $\mathcal{S}$ which simulates the protocol $\mathcal{F}^{\texttt{skup}}$ specification and answers all queries of the adversary.

Definition 4 (Adversarial Queries): The adversary $\mathcal{A}$ participates in the actual execution of the system-key update protocol instance $\mathcal{F}^{\texttt{skup}}$ and may, at any time, send the following queries to an oracle:

- Execute query `Execute(TPD,CA)`: This query models passive attacks. The adversary $\mathcal{A}$ eavesdrops the execution of the system-key update protocol instance $\mathcal{F}^{\texttt{skup}}$ between a chosen TPD and the CA. $\mathcal{A}$ learns the corresponding transcript.
- Send query `Send(m,F)`: This query models active attacks where $\mathcal{A}$ sends a message m to the system-key update protocol instance $\mathcal{F}^{\texttt{skup}}$.
- Hash oracle query `Hash(`m`)`: In this query, $\mathcal{A}$ presents an oracle with $m \in \{0,1\}^*$. The oracle responds to $\mathcal{A}$'s query with the value `Hash(`m`)` $\in \{0,1\}^l$, where l is string length of the hash output.
- Encrypt oracle query `Encrypt(`m_0`,`m_1`)`: $\mathcal{A}$ submits two messages $m_0, m_1 \in \{0,1\}^l$ to an encryption oracle. The oracle responds to $\mathcal{A}$'s query with the encrypted value of one of the messages m_b using a random key $k \in_R \mathbb{Z}_q^*$ where $b \in \{0,1\}$.

B. FORMAL SECURITY PROOF

Claim: The security of $\mathcal{F}^{\texttt{skup}}$ (as defined in Section IV-A) is based on the security of PRF, (Enc,Dec), and HMAC (as defined in Section IV-B) in function of the security parameter λ.

Proof. (Sketch): The ultimate goal of $\mathcal{A}$ is to maliciously commit in the network communications. We construct a sequence of games G_i, $i = 0, \ldots, 8$. The event that adversary $\mathcal{A}$ breaks the security of $\mathcal{F}^{\texttt{skup}}$ in game G_i is denoted by $\texttt{win}_i$.

Game G_0. [Real protocol] This is the real game $\texttt{Real-Game}_{\mathcal{S}}^{\Gamma_{\texttt{2FLIP}}}(\lambda, \mathcal{A})$ played between $\mathcal{A}$ and $\mathcal{S}$ which simulates the protocol $\mathcal{F}^{\texttt{skup}}$ according to the natural protocol specification and answers the adversarial queries.

Game G_1. [Same random secret k_{up} in $\mathcal{F}^{\texttt{skup-req}}$] In this game the simulation aborts if during the interaction the simulator on behalf of the TPD chooses random k_{up} in the vehicle system-key update protocol $\mathcal{F}^{\texttt{skup-req}}$ during TPD $\rightarrow$ CA key-update request. Considering the probability for the collision of two random choices (λ-bit length each) we obtain:

$$|Pr_{\mathcal{A}}[\texttt{win}_1(\mathcal{F}^{\texttt{skup-req}})] - Pr_{\mathcal{A}}[\texttt{win}_0]| \leq \frac{1}{2^{p(\lambda)}}.$$

Game G_2. [Pseudo-randomness of k_1, k_2, and k_3 in $\mathcal{F}^{\texttt{skup-req}}$] In this game the simulator on behalf of the TPD chooses random k_1, k_2, and k_3 instead of computing them using the pseudo random function PRF in the vehicle system-key update protocol $\mathcal{F}^{\texttt{skup-req}}$ during TPD $\rightarrow$ CA key-update request, where the random secret k_{up} was used to compute $k_1 = HKDF(1, k_{up})$ (secret key for encrypting Q_t), $k_2 = HKDF(2, k_{up})$ (secret key for generating HMAC tag), and $k_3 = HKDF(3, k_{up})$ (secret key for generating a HMAC tag as a reference ref). Due to the pseudo-randomness of PRF, we obtain:

$$|Pr_{\mathcal{A}}[\texttt{win}_2(\mathcal{F}^{\texttt{skup-req}})] - Pr_{\mathcal{A}}[\texttt{win}_1]| \leq 3Adv^{\texttt{ind-sec}}_{\mathcal{A}^{(PRF)^{\lambda}}}.$$

Game G_3. [HMAC Forgery in $\mathcal{F}^{\texttt{skup-req}}$] In this game the simulation aborts if $\mathcal{A}$ queries on the message $\{m \parallel {}_{k_2}\delta(m)\}$ in the vehicle system-key update protocol $\mathcal{F}^{\texttt{skup-req}}$ such that ${}_{k_2}\delta(m)$ is a valid HMAC authentication tag on a TPD $\rightarrow$ CA key-update request string $\{m = ref, {}_{k_1}C(Q_t), t_s\}$. To achieve this, $\mathcal{A}$ must find a new message/tag pair, or an old message as long as the output tag was not previously attached to this message by a legitimate TPD. If $\mathcal{A}$ can do this, we can say that $\mathcal{A}$ breaks the SUF-CMA security of the applied HMAC scheme. Thus:

$$|Pr_{\mathcal{A}}[\texttt{win}_3(\mathcal{F}^{\texttt{skup-req}})] - Pr_{\mathcal{A}}[\texttt{win}_2]| \leq Succ^{\texttt{suf-cma}}_{\mathcal{A}^{(HMAC)^\lambda}}.$$

Game G_4. [Indistinguishability of ${}_{k_1}C(Q_t)$ in $\mathcal{F}^{\texttt{skup-req}}$] In this game the simulator replaces the ciphertext ${}_{k_1}C(Q_t)$ with ${}_{k_1}C(y) = \texttt{Encrypt(y, k}_1\texttt{)}$ for randomly chosen y in the vehicle system-key update protocol $\mathcal{F}^{\texttt{skup-req}}$ during TPD $\rightarrow$ CA key-update request. Due to the IND-CPA property of (Enc,Dec) we obtain:

$$|Pr_{\mathcal{A}}[\texttt{win}_4(\mathcal{F}^{\texttt{skup-req}})] - Pr_{\mathcal{A}}[\texttt{win}_3]| \leq Adv^{\texttt{ind-cpa}}_{\mathcal{A}^{(Enc,Dec)^\lambda}}.$$

Game G_5. [Same random secret k' in $\mathcal{F}^{\texttt{skup-res}}$] In this game the simulation aborts if during the interaction the simulator on behalf of the CA chooses the same random secret $k' = b.Q_t$ in the vehicle system-key update protocol $\mathcal{F}^{\texttt{skup-res}}$ during TPD $\leftarrow$ CA key-update response. Considering the probability for the collision of two random choices (λ-bit length each) we obtain:

$$|Pr_{\mathcal{A}}[\texttt{win}_5(\mathcal{F}^{\texttt{skup-res}})] - Pr_{\mathcal{A}}[\texttt{win}_4]| \leq \frac{1}{2^{p(\lambda)}}.$$

Game G_6. [Pseudo-randomness of k'_1 and k'_2 in $\mathcal{F}^{\texttt{skup-res}}$] In this game the simulator on behalf of the CA chooses random k'_1 and k'_2 instead of computing them using the pseudo random function PRF in the vehicle system-key update protocol $\mathcal{F}^{\texttt{skup-res}}$ during TPD $\leftarrow$ CA key-update response, where the random secret $k' = b.Q_t$ was used to compute $k'_1 = HKDF(1, k')$ (secret key for encrypting T'_{skup}) and $k'_2 = HKDF(2, k')$ (secret key for generating HMAC tag). Due to the pseudo-randomness of PRF we obtain:

$$|Pr_{\mathcal{A}}[\texttt{win}_6(\mathcal{F}^{\texttt{skup-res}})] - Pr_{\mathcal{A}}[\texttt{win}_5]| \leq 2Adv^{\texttt{ind-sec}}_{\mathcal{A}^{(PRF)^\lambda}}.$$

Game G_7. [HMAC Forgery in $\mathcal{F}^{\texttt{skup-res}}$] In this game the simulation aborts if $\mathcal{A}$ queries on the message $\{m \parallel {}_{k'_2}\delta(m)\}$ in the vehicle system-key update protocol $\mathcal{F}^{\texttt{skup-res}}$ such that ${}_{k'_2}\delta(m)$ is a valid HMAC authentication tag on a TPD $\leftarrow$ CA key-update response string $\{m = ref, {}_{k'_1}C(T'_{skup}), \beta, CA_{par}, t_s\}$. To achieve this, $\mathcal{A}$ must find a new message/tag pair, or an old message as long as the output tag was not previously attached to this message by a legitimate TPD. If $\mathcal{A}$ can do this, we can say that $\mathcal{A}$ breaks the SUF-CMA security of the applied HMAC scheme. Thus:

$$|Pr_{\mathcal{A}}[\texttt{win}_7(\mathcal{F}^{\texttt{skup-res}})] - Pr_{\mathcal{A}}[\texttt{win}_6]| \leq Succ^{\texttt{suf-cma}}_{\mathcal{A}^{(HMAC)^\lambda}}.$$

Game G_8. [Indistinguishability of ${}_{k'_1}C(T'_{skup})$ in $\mathcal{F}^{\texttt{skup-res}}$] In this game the simulator replaces the ciphertext ${}_{k'_1}C(T'_{skup})$ with ${}_{k'_1}C(y) = \texttt{Encrypt(y, k}'_1\texttt{)}$ for randomly chosen y in the vehicle system-key update protocol $\mathcal{F}^{\texttt{skup-res}}$ during TPD $\leftarrow$ CA response. Note that T'_{skup} contains new key-material. Due to the IND-CPA property of (Enc,Dec) we obtain:

$$|Pr_{\mathcal{A}}[\texttt{win}_8(\mathcal{F}^{\texttt{skup-res}})] - Pr_{\mathcal{A}}[\texttt{win}_7]| \leq Adv^{\texttt{ind-cpa}}_{\mathcal{A}^{(Enc,Dec)^\lambda}}.$$

The probability of $\mathcal{A}$ in winning Game G_8 is:

$$Pr_{\mathcal{A}}[\texttt{win}_8] \leq \begin{cases} 0 \\ 2^{-p(\lambda)}, \end{cases}$$

which yields the result stated as the following theorem. ■

Theorem 1: If PRF is pseudo random, (Enc,Dec) is IND-CPA secure, and HMAC is SUF-CMA secure (as defined in Section IV-B), then $\mathcal{F}^{\texttt{skup}}$ in $\Gamma_{\texttt{2FLIP}}$ (refer to Section IV-D) is secure in the following sense: the maximum probability of success for $\mathcal{A}$ to break the security of $\mathcal{F}^{\texttt{skup}}$ (as defined in Section IV-A) is a negligible function of the security parameter λ as follows:

$$Succ^{\mathcal{F}^{\texttt{skup}}}_{\mathcal{S}}(\lambda, \mathcal{A}) \leq 2\frac{1}{2^{p(\lambda)}} + 5Adv^{\texttt{ind-sec}}_{\mathcal{A}^{(PRF)^\lambda}} + 2Succ^{\texttt{suf-cma}}_{\mathcal{A}^{(HMAC)^\lambda}} + 2Adv^{\texttt{ind-cpa}}_{\mathcal{A}^{(Enc,Dec)^\lambda}}. \quad (1)$$

Corollary 2: The vehicle system-key protocol $\mathcal{F}^{\texttt{skup}}$ provides anonymity and unlinkability in the sense of randomness presented in Games G_1, G_2, G_5, and G_6. That is, $\mathcal{A}$ has a low probability of success in identifying a vehicle TPD among the set of all vehicles using its transmitted data.

Corollary 3: The vehicle system-key protocol $\mathcal{F}^{\texttt{skup}}$ provides confidentiality of new system key in the sense of indistinguishability presented in Games G_8.

VI. PERFORMANCE ANALYSIS

This section evaluates the performance of the new system-key update protocol presented in Section IV-D.

The investigation covers the Average Computation Time (ACT) of the various cryptographic operations providing 128-bit security level, including: Elliptic Curve Cryptography (ECC) base point scalar multiplication (T^{scal}_{mul}), AES-128 CTR mode encryption (T^{aes}_{enc}) and decryption (T^{aes}_{dec}), HKDF-256 (T^{hkdf}_{sha}), and HMAC-256 (T^{hmac}_{sha}). The investigation is performed using the newest stable release branch (1.1.1 series) of the OpenSSL software library [39]. All ECC operations are evaluated over NIST P-256 curve for achieving 128-bit security level [40]. The experiment for each cryptographic operation involved 10^6 trials in Debian Linux distribution running on an Intel Core 2 Duo Processor T6570 (2Megabytes Cache, 2.10 GHz, 4 GB RAM). Note that the latest high-performance automotive single chip modem for vehicular communications made by NXP Semiconductors (the RoadLINK SAF5400 [41]) offers the same performance as the 2.10 GHz Intel Core 2 Duo-T6570 processor used in this study [5], [7], [42].

To have more useful results, this research practically estimates the cryptographic overhead of the new system-key

TABLE 3. The Average Computation Time and Cryptographic Overhead of Different Operations

	$scal_{mul}$	aes_{enc}	aes_{dec}	$hkdf_{sha}$	$hmac_{sha}$
ACT (sec)	1.4×10^{-4}	6.8×10^{-7}	8.9×10^{-7}	3.7×10^{-6}	3.7×10^{-6}
Size (byte)	32	32	32	–	32

TABLE 4. The Computational Cost in Our Key-Update Protocol (For Each Step)

Steps	Computational Overhead (sec)
1. CA ↓	$2T^{hkdf}_{sha} + T^{hmac}_{sha} = 1.11 \times 10^{-5}$
2. TPD → CA	$5T^{hkdf}_{sha} + 2T^{hmac}_{sha} + T^{scal}_{mul} + T^{aes}_{enc} = 1.66 \times 10^{-4}$
3. TPD ← CA	$4T^{hkdf}_{sha} + 2T^{hmac}_{sha} + T^{scal}_{mul} + T^{aes}_{dec} + T^{aes}_{enc} = 1.63 \times 10^{-4}$
4. TPD ↓	$2T^{hkdf}_{sha} + T^{hmac}_{sha} + T^{scal}_{mul} + T^{aes}_{dec} = 1.51 \times 10^{-4}$

TABLE 5. The Communication Overhead in Our Key-Update Protocol (Request and Response)

Steps	Cryptographic Overhead (byte)
1. TPD → CA	$\underbrace{k_3\delta(0)}_{\text{ref}} + \underbrace{k_1 C(Q_t)}_{\text{aes enc}} + \underbrace{t_s}_{\text{time}} + \underbrace{k_2\delta(m)}_{\text{hmac}} = 100$
2. TPD ← CA	$\underbrace{k_3\delta(0)}_{\text{ref}} + \underbrace{k'_1 C(T'_{skup})}_{\text{aes enc}} + \underbrace{\beta}_{\text{ecies param}} + \underbrace{t_s}_{\text{time}} + \underbrace{k'_2\delta(m)}_{\text{hmac}} = 148$

TABLE 6. The Computational Cost in 2FLIP Key-Update Protocol (CA to TPD)

Steps	Computational Overhead (sec)
1. TPD ← CA	$T^{aes}_{enc} + T^{hash}_{sha} + 2T^{ate}_{pairing} \approx 10.8 \times 10^{-3}$
2. TPD ↓	$T^{aes}_{dec} + T^{hash}_{sha} + 2T^{ate}_{pairing} \approx 10.8 \times 10^{-3}$

update protocol on packet size. Table 3 lists the results of investigation.

A. COMPUTATIONAL COST

This section evaluates the computational cost of the new system-key update protocol presented in Section IV-D. We use the evaluated ACT (for each single cryptographic operation in Table 3) to calculate the total computational cost in each step of our proposed system-key update protocol.

For the first step, the CA requires to perform two HKDF and one HMAC calculations (overall 1.11×10^{-5} sec), then sends the PIN to the user. This preparation is only required for the phone-based key-update.

For the second step TPD → CA, the TPD requires to perform five HKDF, two HMAC, one point scalar multiplication (elliptic curve calculation [43]), and one symmetric encryption computations (overall 1.66×10^{-4} sec).

For the third step TPD ← CA, the CA requires to perform four HKDF, two HMAC, one point scalar multiplication, one symmetric decryption, and one symmetric encryption computations (overall 1.63×10^{-4} sec).

For the fourth step, the TPD requires to perform two HKDF, one HMAC, one point scalar multiplication, and one symmetric decryption computations (overall 1.51×10^{-4} sec). Table 4 summarizes the results of investigation. Please refer to Table 3 to find the ACT corresponding to each single operation.

B. COMMUNICATION OVERHEAD

This section evaluates the communication overhead of the new system-key update protocol presented in Section IV-D.

The new system-key update protocol produces 100 bytes of overhead during TPD → CA key-update request, including: one HMAC tag (the reference number) $ref = {}_{k_3}\delta(0) = 32$ bytes, one AES ciphertext ${}_{k_1}C(Q_t) = 32$ bytes (for 32 bytes ECC point Q_t in compressed size), one time stamp $t_s = 4$ bytes, and one HMAC ${}_{k_2}\delta(m) = 32$ bytes. Note that the AES-CTR encrypted ECC point ${}_{k_1}C(Q_t)$ has a size similar to the point Q_t before encryption, for example ${}_{k_1}C(Q_t) = 100$ bytes where $Q_t = 100$ bytes.

The protocol also produces 148 bytes of overhead during TPD ← CA key-update response, including: one HMAC tag $ref = 32$ bytes, one AES ciphertext ${}_{k'_1}C(T'_{skup}) = 48$ bytes (for 32 bytes new system key k'_m and 16 bytes new $salt_u$ in $T'_{skup} = \{k'_m, salt_u\}$), one ECIES parameter (ECC point in compressed size) $\beta = 32$ bytes, one time stamp $t_s = 4$ bytes, and one HMAC ${}_{k'_2}\delta(m) = 32$ bytes. Table 5 summarizes the results of investigation. Please refer to Table 3 to find the cryptographic overhead corresponding to each payload.

VII. COMPARISON AND DISCUSSION

Unlike Wang et al.'s protocol [1], the proposed system-key update protocol in this paper does not use an old system key to encrypt and decrypt a new system key, and as a result, it does not produce dependencies among keying material. In case of compromise of past system key, a new system key is not compromised.

In terms of computational cost, the 2FLIP system-key update protocol requires one symmetric encryption/decryption, one hash, and two cryptographic pairing computations (see [1, §VI – Table 6]). To calculate the computational cost of cryptographic pairing [42], [44], an investigation is performed using the newest stable release branch (1.1.1 series) of the OpenSSL software library [39], as declared in Section VI. The elliptic curve arithmetic and optimal Ate pairing [45] are over the Barreto-Naehrig (BN) curve [46] in the evaluation, where the minimum bit-length of p is estimated as 382 bits (achieving 127-bit security level [47]), an optimistic parameter to use in cryptographic pairings for 128-bit security level (384 bits p) [48].

Table 6 summarizes the results of investigation for computational cost in 2FLIP key-update protocol (CA to TPD). The result for one optimal Ate pairing is 5.4×10^{-3} sec. Ignoring the symmetric decryption and hash calculation, only two cryptographic pairing computations requires 10.8×10^{-3} sec, approximately 70 times slower than our proposed protocol in

TABLE 7. The Communication Overhead in 2FLIP Key-Update Protocol (Broadcast From CA)

Steps	Cryptographic Overhead (byte)
TPD ← CA	$\underbrace{Enc_{k_m}(ts'_{key} \parallel ID_{CA} \parallel k'_m)}_{\text{encrypted time, id, and new key}} + \underbrace{sg}_{\text{id-based signatute}} = 88$

TABLE 8. The Performance Overhead Comparison Between 2FLIP and Our Proposed Key-Update Protocol

Schemes	Computational (sec)	Communication (byte)
2FLIP TPD → CA	*not provided*	*not provided*
2FLIP TPD ← CA	$\approx 10.8 \times 10^{-3}$	88
2FLIP TPD ↓	$\approx 10.8 \times 10^{-3}$	–
Ours TPD → CA	1.66×10^{-4}	100
Ours TPD ← CA	1.63×10^{-4}	148
Ours TPD ↓	1.51×10^{-4}	–

[Footnote 1] Results on an Intel Core 2 Duo Processor T6570 (2M Cache, 2.10 GHz, 4GB RAM, Single Core).

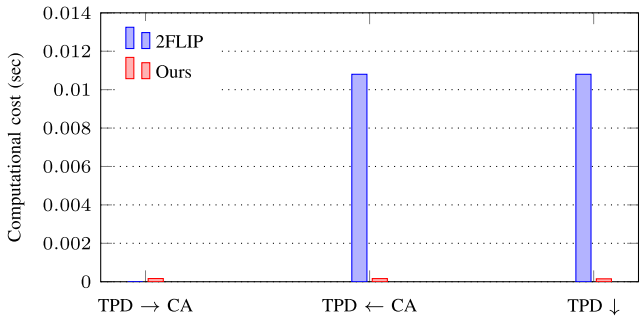

FIGURE 6. The computational cost comparison between 2FLIP and our proposed key-update protocol.

this paper with 1.6×10^{-4} sec computational overhead (refer to Section VI-A).

In terms of communication overhead, the 2FLIP system-key update protocol generates 88 bytes of overhead (see [1, §VI – Table 6]). However, our new protocol presented in this paper generates 148 bytes of overhead (refer to Section VI-B), only 60 bytes more than 2FLIP. Table 7 summarizes the results of investigation for communication overhead in 2FLIP key-update protocol (Broadcast from CA).

In our proposal, the key-update procedure can be called by a vehicle TPD (TPD → CA), or advertised by the CA (TPD ← CA) through RSUs. However, the system-key update request from TPD to CA (TPD → CA) in 2FLIP is a missing functionality. The only entity that can call a system-key update procedure is CA. Table 8 shows the performance overhead comparison between 2FLIP and our proposed system-key update protocol. Please refer to Figs. 6 and 7 to see a detailed comparison for each computational and communication overhead, respectively.

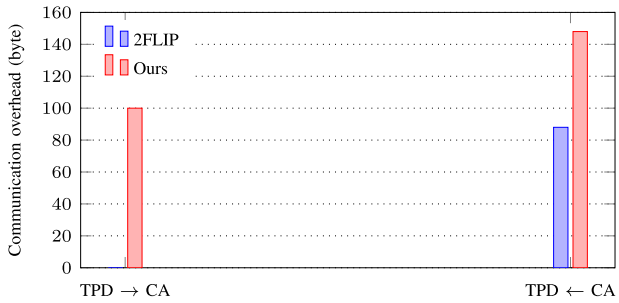

FIGURE 7. The communication overhead comparison between 2FLIP and our proposed key-update protocol.

The new system-key update protocol presented in this paper uses the encrypt-then-MAC method [49] where the encryption scheme is IND-CPA secure (refer to Section IV-B1) and the MAC is SUF-CMA secure (refer to Section IV-B2). This method implies that decryption will only be carried on ciphertext that passed an integrity test. The adversary thus cannot observe use of the decryption key on ciphertext which is chosen randomly, since that ciphertext will not pass the MAC verification and will not be decrypted. Hence, the ECIES used in this protocl should provide Indistinguishability under Chosen-Ciphertext Attacks (IND-CCA) [30]: the adversary has an additional capability over the IND-CPA, calling an encryption or decryption oracle for encrypting or decrypting arbitrary messages before obtaining the challenge ciphertext.

VIII. CONCLUSION AND FUTURE DIRECTION

This paper presents an analysis of Wang et al. [1] *2FLIP: A Two-Factor Lightweight Privacy Preserving Authentication Scheme for VANET*. The 2FLIP authors claimed that their scheme provides a secure system-key update and resists message forgery attacks. However, we demonstrated that this is incorrect. Their scheme could not achieve the claimed goals as it does not provide perfect forward secrecy. We demonstrated this with two successful attacks on 2FLIP: a known-key attack reveals subsequent keys, and following this, a message forgery attack using the known key can be performed. We have proposed a new system-key update protocol that addresses the security flaws in the 2FLIP scheme. We performed an in-depth security analysis supported by formal security proof to demonstrate that our solution can effectively satisfy the security and privacy requirements. We demonstrated the efficiency of our protocol by conducting extensive performance analysis. The enhanced system-key update protocol will also be useful for researchers and designers to apply in VANET authentication schemes.

As quantum computing becomes feasible, many of the current public-key cryptographic schemes will be broken. Most of the existing authentication and key-management protocols in public literature rely on these algorithms. However, to the best of our knowledge, there have been no research efforts that examine the quantum-safety aspects of key-management protocols in the context of vehicular communications. This is a potential avenue for future research.

REFERENCES

[1] F. Wang, Y. Xu, H. Zhang, Y. Zhang, and L. Zhu, "2FLIP: A two-factor lightweight privacy-preserving authentication scheme for VANET," *IEEE Trans. Veh. Technol.*, vol. 65, no. 2, pp. 896–911, Feb. 2016.

[2] B. Hassanabadi and S. Valaee, "Reliable periodic safety message broadcasting in VANETs using network coding," *IEEE Trans. Wireless Commun.*, vol. 13, no. 3, pp. 1284–1297, Mar. 2014.

[3] P. Papadimitratos, A. D. La Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza, "Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation," *IEEE Commun. Mag.*, vol. 47, no. 11, pp. 84–95, Nov. 2009.

[4] CAMP, "Vehicle safety communications project: Task 3 final report - identify intelligent vehicle safety applications enabled by DSRC," *Tech. Rep.*DOT HS 809 859, National Highway Traffic Safety Administration - U. S. Department of Transportation (USDOT), Mar. 2005.

[5] M. A. R. Baee, L. Simpson, E. Foo, and J. Pieprzyk, "Broadcast authentication in latency-critical applications: On the efficiency of IEEE 1609.2," *IEEE Trans. Veh. Technol.*, vol. 68, no. 12, pp. 11577–11587, Dec. 2019.

[6] R. A. Uzcategui, A. J. D. Sucre, and G. Acosta-Marum, "Wave: A tutorial," *IEEE Commun. Mag.*, vol. 47, no. 5, pp. 126–133, May 2009.

[7] M. A. R. Baee, L. Simpson, X. Boyen, E. Foo, and J. Pieprzyk, "A model to evaluate reliability of authentication protocols in C-ITS safety-critical applications," *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 9306–9319, Sep. 2021.

[8] M. A. R. Baee, "Privacy-preserving authentication and key management for cooperative intelligent transportation systems," Ph.D. dissertation, Sch. Comput. Sci., Queensland Univ. of Technol., Brisbane, Australia, 2021.

[9] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*. Boca Raton, FL, USA: CRC Press, Inc., 1st ed., 1996.

[10] S. Goudarzi et al., "A systematic review of security in vehicular ad hoc network," in *Proc. 2nd Symp. Wireless Sensors Cellular Netw.*, 2013, pp. 1–10.

[11] S. A. Soleymani et al., "Trust management in vehicular ad hoc network: A systematic review," *EURASIP J. Wireless Commun. Netw.*, vol. 2015, May 2015, Art. no. 146.

[12] M. Raya and J.-P. Hubaux, "The security of vehicular ad hoc networks," in *Proc. 3rd ACM Workshop Secur. Ad Hoc Sensor Netw.*, New York, NY, USA, 2005, pp. 11–21.

[13] X. Lin, X. Sun, P. Ho, and X. Shen, "GSIS: A secure and privacy-preserving protocol for vehicular communications," *IEEE Trans. Veh. Technol.*, vol. 56, no. 6, pp. 3442–3456, Nov. 2007.

[14] A. Studer, F. Bai, B. Bellur, and A. Perrig, "Flexible, extensible, and efficient VANET authentication," *J. Commun. Netw.*, vol. 11, pp. 574–588, Dec. 2009.

[15] P. Vijayakumar, M. Azees, and L. J. Deborah, "CPAV: Computationally efficient privacy preserving anonymous authentication scheme for vehicular ad hoc networks," in *Proc. IEEE 2nd Int. Conf. Cyber Secur. Cloud Comput.*, 2015, pp. 62–67.

[16] P. Vijayakumar, M. Azees, V. Chang, J. Deborah, and B. Balusamy, "Computationally efficient privacy preserving authentication and key distribution techniques for vehicular ad hoc networks," *Cluster Comput.*, vol. 20, no. 3, pp. 2439–2450, 2017.

[17] J. Huang, Y. Qian, and R. Q. Hu, "Secure and efficient privacy-preserving authentication scheme for 5G software defined vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 8, pp. 8542–8554, Aug. 2020.

[18] J. Camenisch, M. Drijvers, A. Lehmann, G. Neven, and P. Towa, "Zone encryption with anonymous authentication for V2V communication," in *Proc. IEEE Eur. Symp. Secur. Privacy*, 2020, pp. 405–424.

[19] B. Palaniswamy, S. Camtepe, E. Foo, L. Simpson, M. A. Rezazadeh Baee, and J. Pieprzyk, "Continuous authentication for VANET," *Veh. Commun.*, vol. 25, 2020, Art. no. 100255.

[20] X. Li, T. Liu, M. S. Obaidat, F. Wu, P. Vijayakumar, and N. Kumar, "A lightweight privacy-preserving authentication protocol for VANETs," *IEEE Syst. J.*, vol. 14, no. 3, pp. 3547–3557, Sep. 2020.

[21] M. A. Khan, A. Ghani, M. S. Obaidat, P. Vijayakumar, K. Mansoor, and S. A. Chaudhry, "A robust anonymous authentication scheme using biometrics for digital rights management system," in *Proc. Int. Conf. Commun., Comput., Cybersecurity, Inform.*, 2021, pp. 1–5.

[22] B. Gupta, V. Prajapati, N. Nedjah, P. Vijayakumar, A. A. A. El-Latif, and X. Chang, "Machine learning and smart card based two-factor authentication scheme for preserving anonymity in Telecare Medical Information System (TMIS)," *Neural Comput. Appl.*, pp. 1–26, 2021. [Online]. Available: https://link.springer.com/article/10.1007/s00521-021-06152-x#citeas

[23] C. Wang, R. Huang, J. Shen, J. Liu, P. Vijayakumar, and N. Kumar, "A novel lightweight authentication protocol for emergency vehicle avoidance in VANETs," *IEEE Internet Things J.*, vol. 8, no. 18, pp. 14248–14257, Sep. 2021.

[24] M. A. R. Baee, L. Simpson, X. Boyen, E. Foo, and J. Pieprzyk, "Authentication strategies in vehicular communications: A taxonomy and framework," *EURASIP J. Wireless Commun. Netw.*, vol. 2021, no. 1, pp. 1–50, 2021.

[25] M. A. Rezazadeh Baee, L. Simpson, X. Boyen, E. Foo, and J. Pieprzyk, "ALI: Anonymous lightweight inter-vehicle broadcast authentication with encryption," *IEEE Trans. Dependable Secure Comput.*, early access, 2022, doi: 10.1109/TDSC.2022.3164436.

[26] A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem," *Proc. London Math. Soc.*, vol. 2, no. 1, pp. 230–265, 1937.

[27] M. Abdalla, M. Bellare, and P. Rogaway, "DHAES: An encryption scheme based on the diffie-hellman problem," *IACR Cryptol. ePrint Arch.*, vol. 1999, 1999, Art. no. 7.

[28] M. Just, *Diffie–Hellman Key Agreement*. Boston, MA, USA: Springer, 2005, pp. 154–154.

[29] M. Dworkin, "Recommendation for block cipher modes of operation. methods and techniques," Tech. Rep., *Nat. Inst. of Standards and Technol. Gaithersburg MD Comput. Secur. Div*, 2001.

[30] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2000, pp. 531–545.

[31] P. J. Bond, "The keyed-hash message authentication code (HMAC)," *Federal Inf. Process. Standards Publ.*, vol. 198, pp. 1–21, 2002.

[32] Q. Dang, "Changes in federal information processing standard (FIPS) 180-4, secure hash standard," *Cryptologia*, vol. 37, no. 1, pp. 69–73, 2013.

[33] H. Krawczyk, "Cryptographic extraction and key derivation: The HKDF scheme," in *Proc. Annual Cryptology Conference*, 2010, pp. 631–648.

[34] D. H. Krawczyk and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)," RFC 5869, May 2010.

[35] M. Bellare, "New proofs for NMAC and HMAC: Security without collision resistance," *J. Cryptol.*, vol. 28, no. 4, pp. 844–878, 2015.

[36] B. Kaliski, "PKCS #5: Password-Based Cryptography Specification Version 2.0.," RFC 2898, Sep. 2000.

[37] L. C. Washington, *Elliptic Curves: Number Theory and Cryptography*. London, U.K.: Chapman and Hall/CRC, 2008.

[38] O. Goldreich, *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge, U.K.: Cambridge Univ. Press, 2007.

[39] *The OpenSSL Project*, "OpenSSL: The open source toolkit for SSL/TLS," Apr. 2003. [Online]. Available: www.openssl.org

[40] IEEE, "IEEE standard for wireless access in vehicular environments–security services for applications and management messages - Amendment 1," IEEE Std 1609.2a-2017 (Amendment to IEEE Std 1609.2-2016), pp. 1–123, Oct. 2017.

[41] *NXP Semiconductors; RoadLINK SAF5400 single chip modem*. Accessed: May 10, 2022. [Online]. Available: https://www.nxp.com

[42] M. A. R. Baee, L. Simpson, X. Boyen, E. Foo, and J. Pieprzyk, "On the efficiency of pairing-based authentication for connected vehicles: Time is not on our side!," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 3678–3693, Jun. 2021.

[43] I. F. Blake, G. Seroussi, and N. P. Smart, *Elliptic Curves in Cryptography*. New York, NY, USA: Cambridge. U.K.: Cambridge Univ. Press, 1999.

[44] M. A. R. Baee, "Implementation and performance analysis of identity-based authentication in wireless sensor networks," Master's thesis, Fac. Comput., Universiti Teknologi Malaysia, Johor Bahru, Malaysia, 2014.

[45] F. Vercauteren, "Optimal pairings," *IEEE Trans. Inf. Theory*, vol. 56, no. 1, pp. 455–461, Jan. 2010.

[46] P. S. L. M. Barreto and M. Naehrig, "Pairing-friendly elliptic curves of prime order," in *Proc. Int. Workshop Sel. Areas Cryptography*, Berlin, Heidelberg, 2006, pp. 319–331.

[47] Y. Sakemi, T. Kobayashi, and T. Saito, "Pairing-friendly curves," Internet-Draft draft-irtf-cfrg-pairing-friendly-curves-00, Internet Engineering Task Force, Nov. 2019.
[48] A. Menezes, P. Sarkar, and S. Singh, "Challenges with assessing the impact of NFS advances on the security of pairing-based cryptography," in *Proc. Int. Conf. Cryptol. Malaysia*, 2017, pp. 83–108.
[49] P. Gutmann, "Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)," RFC 7366, Sep. 2014.

MIR ALI REZAZADEH BAEE (Senior Member, IEEE) received the Ph.D. degree in computer science information security from the Queensland University of Technology, Brisbane, QLD, Australia. He has more than 15 years of experience working in computer science, and since 2012, he has been involved in computer science research with a strong focus on applied cryptography and information security. His contributions have led to novel and important scientific outcomes. He is an Information Security Researcher with the Queensland University of Technology, collaborating with the Cyber Security Cooperative Research Center, Australia, to solve pressing real-world cyber security challenges. He is a member of the International Association for Cryptologic Research. He was actively a Reviewer for flagship journals such as the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, *Internet of Things*, and conferences including the IACR's *EUROCRYPT and ASIACRYPT*.

LEONIE SIMPSON received the Ph.D. degree from the Queensland University of Technology, Brisbane, QLD, Australia, in 2000. She has been involved with information security research for more than 20 years. She has extensive experience analyzing cryptographic algorithms and finding weaknesses that reduce the security provided. She has applied her knowledge of design flaws in algorithms to help develop more secure ciphers, working in teams with Australian and international researchers. She is currently an an Associate Professor and Information Security Researcher with the Queensland University of Technology. She is currently researching efficient authenticated encryption methods for use in securing data transmissions between small, low-power devices in the rapidly growing Internet of Things. Her main research interests include symmetric cryptology, widely used for data protection.

ERNEST FOO (Member, IEEE) received the Ph.D. degree from the Queensland University of Technology, Brisbane, QLD, Australia, in 2000. He is currently an Associate Professor with School of Information and Communication Technology, the Griffith University, Brisbane. From 2007 to 2019, he has been a Senior Lecturer and Researcher with the Information Security Discipline, Queensland University of Technology. He has authored or coauthored more than 90 refereed papers including 20 journal papers. His research interests include the field of secure network protocols with an active interest in the security of industrial controls systems employing machine learning and data mining as well as cryptographic protocols and network simulations.

JOSEF PIEPRZYK is currently a Senior Principal Research Scientist with the Commonwealth Scientific and Industrial Research Organization, Data61, Sydney, NSW, Australia, a Professor with Institute of Computer Science, Polish Academy of Sciences, and an Adjunct Professor with the Queensland University of Technology, Brisbane, QLD, Australia. He has authored or coauthored five books, edited ten books (conference proceedings), six book chapters, and more than 300 papers in refereed journals and refereed international conferences. His main research interests include cryptology and information security. He is the Member of the Editorial Boards for *International Journal of Information Security* (Springer), *Journal of Mathematical Cryptology* (De Gruyter), *Open Access Journal of Cryptography* (MDPI), *International Journal of Applied Cryptography* (Inderscience Publishers), *Fundamenta Informaticae* (IOS Press), *International Journal of Security and Networks* (Inderscience Publishers), and *International Journal of Information and Computer Security* (Inderscience Publishers).