Privacy-Preserving Aggregation-Authentication Scheme for Safety Warning System in Fog-Cloud Based VANET

Yafang Yang, Lei Zhang, *Member, IEEE*, Yunlei Zhao, Kim-Kwang Raymond Choo, *Senior Member, IEEE*, and Yan Zhang

***Abstract*— As cities become smarter, the importance of vehicular ad hoc networks (VANETs) will be increasingly pronounced. To support latency- and time-sensitive applications, there have been attempts to utilize fog-cloud computing in VANETs. There are, however, a number of limitations in existing fog-cloud based VANET deployments, ranging from computation and communication bottlenecks to privacy leakage to costly certificate/pseudonym management to key escrow, and so on. Therefore, in this paper we propose a privacy-preserving aggregation authentication scheme (PPAAS). The scheme is designed for deployment in a safety warning system for fog-cloud based VANETs. Specifically, the PPAAS scheme is realized using a novel efficient anonymous certificateless aggregation signcryption scheme (CASS) proposed in this paper, and allows a fog node to aggregate signcrypted traffic-related messages from surrounding vehicles into an aggregated ciphertext and unsigncrypt them in a batch. We then evaluate the security of PPAAS and demonstrate that it supports confidentiality, authentication, and (efficient) conditional privacy, and key escrow freeness. In particular, our scheme is the first in the literature to achieve efficient conditional privacy, which avoids the need for costly pseudonym management. We also demonstrate that the scheme is practical, based on our simulation results.**

***Index Terms*— Sender anonymity, key escrow freeness, vehicle ad hoc networks, fog computing, safety warning system.**

I. Introduction

VEHICULAR ad hoc networks (VANETs) are not new, but they remain a topic of ongoing interest to both the research community and the broader society (e.g., federal, state, and county governments). This is not surprising due to their potential application in a broad range of applications ranging from smart intelligent / traffic system to smart city, and so on. Generally, a VANET consists of participating vehicles (e.g., unmanned ground and aerial vehicles) and roadside units (RSUs; generally deployed on roads, buildings and other infrastructure installations). Each vehicle is typically equipped with a tamper-proof on-board unit (OBU), which allows the vehicle to collect data in real-time, for example from various onboard sensors. OBUs can also facilitate (wireless) communication between vehicles, RSUs, and/or other Internet connected things in the vicinity or network. For example, VANETs using dedicated short-range communication (DSRC) [1]–[3] can support vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications (e.g., sending and receiving of traffic-related messages between vehicles in V2V communications, or sending and receiving of traffic-related messages in V2I communications).

In a typical VANET setup, a large number of traffic-related messages (e.g., speed, location, and event of interest) will be generated by the vehicles. For instance, in DSRC communications, each vehicle generates and broadcasts a status message every 100-300 ms. Safety warning system is an important safety-related application in VANETs, where traffic-related messages will need to be collected and analyzed, for example in a cloud server [4]–[6]. To mitigate limitations due to bandwidth and timeliness associated with data transfer in cloud-based safety warning system, one can move the computations closer to the data source; hence, the need for fog computing [7]. This is also the focus of this paper (i.e., safety warning system in fog-cloud based VANETs).

A safety warning system in fog-cloud based VANETs typically adopts a three-layer architecture [8], [9], namely: the mobile layer, the fog layer, and the cloud layer.

Manuscript received June 19, 2021; revised October 28, 2021 and December 9, 2021; accepted December 23, 2021. Date of publication January 6, 2022; date of current version January 21, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant U1536205, Grant 61472084, Grant 61972094, Grant 62032005, Grant 61972159, and Grant 61572198; in part by the Open Research Fund of Engineering Research Center of Software/Hardware Co-design Technology and Application, Ministry of Education (East China Normal University); in part by the Fundamental Research Funds for the Central Universities; in part by the National Key Research and Development Program of China under Grant 2017YFB0802000; in part by the Shanghai Innovation Action Project under Grant 16DZ1100200; in part by the Shanghai Science and Technology Development Funds under Grant 16JC1400801; in part by the Technical Standard Project of Shanghai Scientific and Technological Committee under Grant 21DZ2200500; and in part by the Shandong Provincial Key Research and Development Program of China under Grant 2017CXG0701 and Grant 2018CXGC0701. The work of Kim-Kwang Raymond Choo was supported only by the Cloud Technology Endowed Professorship. The associate editor coordinating the review of this manuscript and approving it for publication was Mr. Frederik Armknecht. *(Corresponding authors: Lei Zhang; Yunlei Zhao.)*

Yafang Yang is with the College of Computer Science and Technology, Fudan University, Shanghai 200433, China (e-mail: 18110240046@fudan.edu.cn).

Lei Zhang and Yan Zhang are with the Engineering Research Center of Software/Hardware Co-design Technology and Application, Ministry of Education, East China Normal University, Shanghai 200062, China, and also with the Shanghai Key Laboratory of Trustworthy Computing, Software Engineering Institute, East China Normal University, Shanghai 200062, China (e-mail: leizhang@sei.ecnu.edu.cn; 1985921943@qq.com).

Yunlei Zhao is with the College of Computer Science and Technology, Fudan University, Shanghai 200433, China, and also with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China (e-mail: ylzhao@fudan.edu.cn).

Kim-Kwang Raymond Choo is with the Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249, USA (e-mail: raymond.choo@fulbrightmail.org).

Digital Object Identifier 10.1109/TIFS.2022.3140657

The mobile layer is composed of stationary and moving vehicles (potentially up to 80 miles per hour or more). The fog layer comprises fog nodes (e.g., RSUs), and the cloud layer is generally formed by one or more cloud servers. In other words, vehicles in the mobile layer will send traffic-related messages to the fog nodes in the fog layer. The fog nodes will process these messages and generate the relevant response messages. Since fog nodes generally have limited storage space, traffic-related messages are forwarded to the cloud layer for storage and/or further processing.

There are a number of security and privacy implications in the design and implementation of any VANET systems, including the VANET-enabled safety warning systems. First, traffic-related messages are transmitted to the fog nodes through open networks and thus, such communications can be subject to attacks with real-world consequences (e.g., modification and/or sending of fabricated messages can result in accidents and/or fatalities). This reinforces the importance of message authentication [13]. Second, message confidentiality is also crucial since the transmitted data may contain sensitive information. In addition, the recipient fog nodes should learn only the content of the messages on a need-to-know basis. Further complicating the system design, the cloud server may also not be fully trusted [14]. Therefore, data stored in the cloud server should be encrypted. Third, a vehicle owner usually does not want his/her personal information (e.g., actual identity and routes) to be revealed. Thus, vehicle (owner) privacy should also be guaranteed. However, we should not rule out that vehicles may fabricate messages and when this occurs, there should be a mechanism to identify and penalize the offending vehicle (owner). Therefore, privacy of the vehicle (owner) should be conditional [15].

Signcryption is a viable solution, since it allows one to achieve both message authentication and message confidentiality simultaneously. Further, the pseudonym mechanism can be utilized to protect the privacy of a vehicle. However, such an approach requires that the vehicle changes its pseudonyms (e.g., anonymous certificates) frequently, which incurs significant pseudonym management costs [16]. One can consider utilizing anonymous signcryption to protect the privacy of a message generator, and thus minimizes the use of pseudonyms. However, efficiency is a real issue that cannot be ignored when designing a (anonymous) signcryption scheme in applications such as safety warning systems in fog-cloud based VANETs, since an RSU may receive a great number of traffic-related messages in a short period of time (e.g., during congested traffic situation) [17]. To verify messages one at a time will be too time-consuming, and can lead to traffic accidents. Furthermore, if a RSU transmits all messages to the cloud server without compression, significant communication and storage overheads are incurred. Hence, we posit the potential of signcryption, since it supports aggregation. This allows one to reduce verification/computation and communication costs, without affecting the security guarantee [18].

Most of the existing signcryption schemes are designed in the conventional public key infrastructure (PKI)-based cryptosystem, and thus incur significant certificate management overheads. Identity-based public key cryptosystem, on the other hand, avoids the need for certificate management but it has the key escrow problem. Hence, we suggest using signcryption schemes designed for the certificateless public key cryptosystem (CPKC), which mitigates the challenges due to certificate management and key escrow. This motivates the design of our proposed anonymous certificateless aggregate signcryption scheme (CASS) in this paper, where CASS is designed to support both sender anonymity and aggregation (two key requirements in safety warning systems). Specifically, CASS consists of a key generation center (KGC), which issues partial private keys for participating entities (e.g., RSUs and vehicles) in the system. This allows the entity to generate the associated public/private key pair based on its partial private key and a self-chosen secret value. With the full private key, the entity can signcrypt messages or unsigncrypt ciphertexts (i.e., signcrypted messages).

To address the above mentioned challenges, a new anonymous CASS is proposed. Building on the proposed anonymous CASS, we also present a privacy-preserving aggregation-authentication scheme (PPAAS). Specifically, we consider the contributions of this paper to be as follows:

1) We propose a novel anonymous CASS, designed to simultaneously support data confidentiality, unforgeability, and sender anonymity. To the best of our knowledge, this is the first provably secure CASS that achieves sender anonymity. In addition, our anonymous CASS supports ciphertext aggregation and batch verification, which makes it more suitable for bandwidth-limited and time-critical applications. Anonymity is one of the basic security requirements of VANETs and a number of CASS has been proposed for VANETs. However, the existing CASSs have not studied the security requirement of sender anonymity.
2) We then use the proposed anonymous CASS, a secure signature scheme, and the pseudonym mechanism as building blocks to design our PPAAS. Subsequently, we evaluate the security of PPAAS to demonstrate that it achieves data confidentiality, authentication, (efficient) conditional privacy and key escrow freeness. We remark that data confidentiality, authentication and key escrow in our scheme are guaranteed by the data confidentiality and unforgeablity of the underlying anonymous CASS (as well as the characteristic of the underpinning CPKC). For (efficient) conditional privacy (see Section III-B), this is realized based on the anonymity property of our CASS and the pseudonym mechanism. We emphasize that this is the first scheme to achieve efficient conditional privacy, and our scheme significantly reduces the costs associated with pseudonym management and storage for vehicles.

The rest of this paper is organized as follows. The next two sections will present related literature and background materials. The proposed CASS and PPAAS are described in Sections IV and V, respectively. Sections VI presents the simulation results of PPAAS and the efficiency and security comparison among our scheme with related schemes. The comparative summary shows our PPAAS incurs low aggregation and unsigncryption delay, and low loss rate. In other

words, the RSU in our scheme can efficiently process received messages; thus, PPAAS can be deployed in real-world applications. The security proof of PPAAS is given in Section VII. Finally, we conclude this work in Section VIII.

II. Related Work

Integrating fog and cloud computing with VANETs is a relatively recent development trend, since such an integrated system will minimize latency and increase bandwidth utilization in time-critical applications [19]. To date, significant research efforts have focused on applications, architecture, practicability of fog-cloud based VANETs [20]–[22]. Generally, existing approaches use a three-layer architecture that is similar to the architecture considered in this paper, with the exception of fog layer composition. For example, a fog layer is composed only of RSUs in [21], and both stationary and moving vehicles are treated as fog nodes in [20], [22]. While in this paper, we describe the fog layer to comprise only RSUs (for simplicity), other devices can also act as fog nodes in practice.

V2V and V2I communications underpin safety warning systems, and there have been a large number of safety warning systems designed based on V2V communications [23]–[25]. However, security and privacy issues are relatively understudied [24], [25]. Further, these schemes are achieved in either PKI-based cryptosystem or IBPKC, or cannot guarantee data confidentiality. We note that, since the communication range of an RSU is much larger than that of a vehicle [26] and multiple RSUs are connected through a wired link, these RSUs may cooperatively analyze the received messages and distribute the resulting safety warning messages to the vehicles in a wide area. This will significantly improve the profundity and precision of the safety warning messages. Only a few researches aim to design safety warning system based on V2I communications [8], [27]. Further, concrete solution is only given in [8] to address the security and privacy issues for safety warning system in fog-cloud based VANET. Although the scheme is achieved in CPKC, it involves a large number of costly cryptographic operations and cannot guarantee user privacy.

The European Telecommunications Standards Institute (ETSI) released a series of standards, ETSI TS 101 539-1, ETSI TS 101 539-2, and ETSI TS 101 539-3, and defined a number of applications based on V2X communications [28]–[30]. One example is the system defined in ETSI TS 101 539-3 for collision risk warning from road side ITS-S (or RSU in the context of this paper), which is conceptually similar to the safety warning system presented in this paper, in which the road side ITS-S broadcasts warning messages generated from the messages collected from nearby vehicles. Traditional PKI based cryptosystem was suggested as a mechanism to guarantee the security of the system. Further, the standards released by ETSI only discuss traditional privacy requirement while efficient conditional privacy is not studied. As previously discussed, such an approach results in expensive certificate/pseudonym management. We note that this system adopts a typical fog-based VANET framework. However, fog nodes usually have limited storage space. Therefore, traffic-related messages should be forwarded to the cloud server for storage and further processing, for example to inform long-term city planning. These are the limitations we seek to address in our proposal.

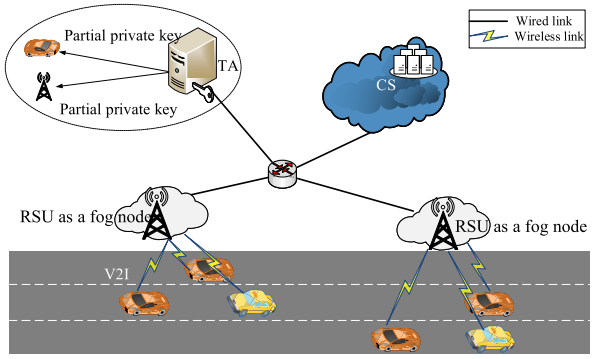


Fig. 1. System Architecture.

Aggregate signcryption is a useful tool that can be used to realize secure communications for safety warning systems in fog-cloud VANETs. Selvi *et al.* [31] proposed the first aggregate signcryption scheme in IBPKC. To eliminate the key escrow problem, the concept of CASS is introduced in [32] together with a concrete construction. Later, some efficient constructions are proposed [8], [33], [34]. However, the number of costly bilinear map operations involved in [8], [33] and [34] increases linearly as the number of vehicles grows. Recently, Wang *et al.* [9] proposed a CASS for the VANETs, which can achieve constant number of bilinear map operations. Nevertheless, this scheme results in huge communication overhead. We note that there are also several CASSs [35], [36] whose security is proven based on the Forking Lemma [37]. However, it was shown that the Forking Lemma is not applicable to the proof of such a scheme [37]. A concrete instance in [37] shows that an attacker can violate the unforgeability of such a scheme. To our knowledge, the above-mentioned CASSs haven't considered the property of sender anonymity. Therefore, it is necessary to design an efficient anonymous CASS for safety warning system in fog-cloud based VANET.

III. Background

A. System Model

Fig. 1 shows our system architecture, in which there exist four types of entities, namely, TA, vehicles, RSUs (i.e., fog nodes), and CS.

- **TA**: The TA is trusted and serves as the KGC. It generates the system parameters and its master secret key, issues partial private key for the entities (i.e., RSUs and vehicles) in the system, and is used to trace the real identity of a vehicle if the vehicle is malicious.
- **Vehicles**: Each vehicle is equipped with a tamper-proof on-board unit (OBU), with which a vehicle can collect its real-time traffic messages and broadcast signcrypted safety-related messages.

- **RSUs**: RSUs are widely distributed in different areas and act as fog nodes [38], [39]. They are connected to each other via wired link. An RSU will collect signcrypted messages broadcasted by the nearby vehicles. These messages are aggregated and used to generate a safety warning message. Finally, the aggregated signcrypted messages will be forwarded to the CS. We note that the RSU can also share the messages with its nearby RSUs, so that they can cooperatively generate a safety warning message.
- **CS**: The CS serves as a data center of the system. All the aggregated signcrypted messages will be stored by the CS and utilized later.

B. Threats and Security Requirements

Security threats in safety warning systems come from both external and internal attackers. Generally, internal threats mainly come from RSUs, vehicles and the CS. The CS and RSUs are usually assumed to be semi-trusted [10]. That means they will honestly perform the PPAAS, but may be interested in drivers' privacy information. Vehicles may be malicious. They may be curious about the content of the messages broadcasted by the surrounding vehicles and/or the identities of the surrounding vehicles, and may also broadcast fake messages that may even cause serious accidents. The threats from an external attacker are similar to those from malicious vehicles. The attacker may be curious about the content of the messages broadcasted by the surrounding vehicles and/or the identities of the surrounding vehicles, and may also impersonate some legal vehicles to send fake messages [40]–[42].

A secure scheme for safety warning system in fog-cloud based VANET has to deal with the above threats. Correspondingly, our scheme is designed to achieve the following security requirements:

- **Data Confidentiality**: The message sent by a vehicle may contain sensitive information. However, an attacker may be curious about the content of the message. Confidentiality guarantees that only the legitimate receivers can learn the content of the message.
- **Authentication**: It guarantees that a received message is indeed generated by the claimed vehicle and unmodified during transmission.
- **Efficient conditional privacy**: It requires that no malicious vehicle or external attacker can extract the sender's real identity or distinguish whether two messages are from the same sender. If this requirement is missing, a malicious vehicle or external attacker can extract a sender's location history. Since there may be malicious vehicles, privacy should be conditional. Specifically, the TA should be able to trace the real identity of a vehicle when it is necessary (e.g., a fake message sent by a malicious vehicle is found). We note that, RSUs in VANETs are usually assumed to be semi-trusted [10]. However, since an RSU is located on the roadside, it has the risk of being corrupted by an attacker. In this case, obviously, the attacker can violate confidentiality. But the privacy of a vehicle should still be protected. This can be achieved by using short-term pseudonyms which results in the pseudonym management problem, since a vehicle has to change its pseudonym frequently. Efficient conditional privacy guarantees that a vehicle only needs to change pseudonym in a long period of time.
- **Key escrow freeness**: Except for the entity (i.e., a vehicle or an RSU) itself, no one else (even the TA) can learn the full private key of the entity.

Since a safety warning system in VANET has to meet the real-time demand, the designed scheme should have low computation and communication overheads in addition to the above security requirements.

C. Anonymous CASS

CPKC aims to eliminate the certificate management problem without suffering from the key escrow problem. In CPKC, a semi-trusted KGC is employed to issue partial private key for the entities (e.g., vehicles, RSUs in this paper) in the system. The full private key of an entity is generated based on its partial private key and a secret value chosen by itself. Since the KGC does not have the secret value, it cannot learn the full private key as well. By this way, the key escrow problem is eliminated. The full public key of an entity in the system is its identity combined with a public key generated using its secret value. However, no certificate is required to bind the identity with the public key, so that the certificate management problem is eliminated.

Signcryption is a cryptographic tool that enables both authentication and confidentiality, where authentication guarantees that a message is indeed sent by the claimed sender without modification and confidentiality guarantees that no one other than the receiver of a message can learn the content of the message. An anonymous signcryption scheme is a signcryption scheme that holds sender anonymity, which guarantees that only a designated receiver of a message can learn the identity of its sender. Anonymous aggregate signcryption, i.e., anonymous signcryption that supports aggregation operation, can aggregate n ciphertexts from n senders into a single ciphertext whose length is much shorter than that of the n ciphertexts. Anonymous CASS is an anonymous aggregate signcryption scheme designed in CPKC, which can guarantee authentication, confidentiality and sender anonymity simultaneously along with the advantage of CPKC.

D. Bilinear Maps and Intractable Problem

Our proposed scheme is from bilinear map which is defined as follows. Let $\mathbb{G}_1$, $\mathbb{G}_2$ be two additive groups of order q (a large prime), $\mathbb{G}_T$ be a multiplicative groups with the same order, P be a generator of $\mathbb{G}_1$, and Q be a generator of $\mathbb{G}_2$. A map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is said to be a bilinear map if it satisfies [43] 1) *Bilinearity*: For all $a, b \in_R \mathbb{Z}_q^*$, $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$. 2) *Non-degeneracy*: For any $u \in \mathbb{G}_1^*$ and $v \in \mathbb{G}_2^*$, $\hat{e}(u, v) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ denotes the identity of $\mathbb{G}_T$.

If $\mathbb{G}_1 = \mathbb{G}_2$, then the map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ is a Type 1 bilinear map; else if there is an efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ and $\psi(Q) = P$, then the map is

TABLE I
NOTATIONS

Notation	Explanation
$\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$	Cyclic groups
$\hat{e}$	A bilinear map from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$
P	A generator of $\mathbb{G}_1$
Q	A generator of $\mathbb{G}_2$
q	The order of $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$ (a large prime)
ψ	A computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$
msk	The master secret key
P_{pub}	The system public key
$H_1 \sim H_6$	Cryptographic hash functions
$P_{ID_i}/D_{ID_i}/sk_{ID_i}$	The public key/partial private key/full private key of an entity with identity ID_i
Δ	An aggregate keyword
RID_{R_i}/RID_{V_i}	The real identity of an RSU/Vehicle
$E_K(\cdot)/D_K(\cdot)$	A secure symmetric key encryption scheme
$\Sigma_{sk}(\cdot)/V_{pk}(\cdot)$	A secure signature scheme
$\oplus$	The bitwise XOR operation
$\|$	The message concatenation operation

a Type 2 bilinear map; else if no such isomorphism exists, the map is called a Type 3 bilinear map [44].

The security of our scheme is based on the assumption that the computational co-Computing-Deffie-Hellman (co-CDH) problem in $\mathbb{G}_1$ is intractable.

Definition 1 (co-CDH assumption): Given a tuple (P, aP, Q, bQ), the co-CDH problem is to calculate abP for unknown $a, b \in \mathbb{Z}_q^*$. The corresponding assumption is that the advantage for any probabilistic polynomial time (PPT) adversary to solve the co-CDH problem is negligible.

E. Notations

Some frequently used notations in our anonymous CASS and PPAAS are shown in Table I.

IV. Building Block

We define the notion and the formal security model of anonymous CASS, then propose a concrete construction that is provably secure in the defined model.

A. Definition of CASS

An anonymous CASS consists of the following algorithms:

- Setup(κ): It is run by the KGC. On input a security parameter κ, it generates the system parameters $para$ and the system master secret key msk. We leave $para$ as a default input of the rest algorithms.
- PPK(msk, ID_i): It is run by the KGC. On input msk, an entity's identity ID_i, it returns a partial private key D_i for the entity.
- UKG(D_i): It is run by an entity with identity ID_i. On input D_i, it returns the full private key sk_{ID_i} and the corresponding public key P_{ID_i} of the entity. We note that sk_{ID_i} and P_{ID_i} are generated based on a secret value chosen by the entity itself.
- SignCrypt($m_i, \Delta, ID_i, P_{ID_i}, sk_{ID_i}, ID_j, P_{ID_j}$): It is run by an entity to signcrypt a message. On input a message m_i, an aggregate keyword Δ, sender's identity ID_i, public key P_{ID_i} and private key sk_{ID_i}, and receiver's identity ID_j and public key P_{ID_j}, it returns a ciphertext C_i, i.e., a signcrypted message.
- AggUnSignCrypt($\{C_i\}_{i=1}^n, \Delta, ID_j, P_{ID_j}, sk_{ID_j}$): It is an aggregation and unsigncryption algorithm run by a receiver to aggregate the received ciphertexts into an aggregated ciphertext and unsigncrypt the ciphertexts. On input a set of ciphertexts $\{C_i\}_{i=1}^n$ generated under the same aggregate keyword Δ, ID_j, P_{ID_j} and sk_{ID_j}, it returns an aggregated ciphertext C, and a set of messages $\{m_i\}_{i=1}^n$ if $\{C_i\}_{i=1}^n$ can pass the verification test.

B. The Construction

We propose our CASS which has the following algorithms:

- Setup(κ): On input κ, the KGC generates the system parameters and master secret key as follows:
  1) Choose $P, Q, q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, a Type 2 bilinear map $\hat{e}: \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ as defined in Section III-D.
  2) Choose $\lambda \in_R \mathbb{Z}_q^*$ as the master secret key, set $P_{pub} = \lambda Q$ as the system public key, choose six cryptographic hash functions: $H_1: \{0,1\}^* \to \mathbb{G}_1^*$, $H_2 \sim H_4: \{0,1\}^* \to \mathbb{G}_2^*$, $H_5: \mathbb{G}_2^3 \to \{0,1\}^k$, $H_6: \{0,1\}^* \to \mathbb{Z}_q^*$, where $k = |m| + |ID|$.
  3) Return the public parameters $para = \langle \mathbb{G}_1, \mathbb{G}_2, \psi, \mathbb{G}_T, \hat{e}, P, Q, P_{pub}, H_1 \sim H_6 \rangle$ and keep λ secret.
- PPK(λ, ID_i): On input msk and ID_i, the KGC computes $Q_{i,0} = H_1(ID_i, 0)$, $Q_{i,1} = H_1(ID_i, 1)$, and sets $D_{ID_i} = (D_{ID_i,0} = \lambda Q_{i,0}, D_{ID_i,1} = \lambda Q_{i,1})$ as the partial private key corresponding to ID_i.
- UKG(D_i): On input D_i, an entity randomly selects $x_{ID_i} \in_R \mathbb{Z}_q^*$, sets $P_{ID_i} = x_{ID_i} P$ as its public key and sets $sk_{ID_i} = (x_{ID_i}, D_i)$ as its full private key.
- SignCrypt($m_i, \Delta, ID_i, P_{ID_i}, sk_{ID_i}, ID_j, P_{ID_j}$): In our later PPAAS, we will design an authentication mechanism to pre-verify the validity of a receiver's public key (See Section V-D). Here we assume the public key of a receiver is already verified. In other words, the public key of the receiver is not replaced. Given m_i, Δ, ID_i and ID_j as input, the sender signcrypt the message m_i as follows:
  1) Select $r_i \in_R \mathbb{Z}_q^*$, set $R_i = r_i P$, $U_i = r_i P_{ID_j}$, set $K_i = H_5(U_i, R_i, P_{ID_j})$, $c_i = K_i \oplus (m_i \| ID_i \| P_{ID_i})$.[1]
  2) Compute $h_i = H_6(P_{ID_i} \| P_{ID_j} \| ID_j \| R_i \| \Delta \| m_i)$, $T = H_2(\Delta)$, $V = H_3(\Delta)$, $W = H_4(\Delta)$.
  3) Compute $s_i = D_{i,0} + x_{ID_i}\psi(V) + h_i(D_{i,1} + x_{ID_i}\psi(W)) + r_i\psi(T)$.
  4) Return $C_i = (R_i, c_i, s_i)$ as the ciphertext.
- AggUnSignCrypt($\{C_i\}_{i=1}^n, \Delta, ID_j, P_{ID_j}, sk_{ID_j}$): On input C, Δ, ID_j, P_{ID_j} and sk_{ID_j}, the receiver does the following:
  1) Compute $S = \sum_{i=1}^n s_i$.
  2) Compute $T = H_2(\Delta), V = H_3(\Delta), W = H_4(\Delta)$; for $1 \le i \le n$, compute $U_i = x_{ID_j} R_i$, $K_i = H_5(U_i, R_i, P_{ID_j})$, $m_i \| ID_i \| P_{ID_i} = K_i \oplus c_i$, $h_i = H_6(P_{ID_i} \| P_{ID_j} \| ID_j \| R_i \| \Delta \| m_i)$, $Q_{i,0} = H_1(ID_i, 0)$, $Q_{i,1} = H_1(ID_i, 1)$.

[1] We note that in existing anonymous CASS, the sender's public key should be transmitted in public. This will reveal some of the sender's identity information. To achieve sender anonymity, the sender's public key should also be encrypted.

3) Verify $\hat{e}(S, Q) \stackrel{?}{=} \hat{e}(\sum_{i=1}^{n} Q_{i,0} + \sum_{i=1}^{n} h_i Q_{i,1}, P_{pub})\ \hat{e}(\sum_{i=1}^{n} P_{ID_i}, V)\hat{e}(\sum_{i=1}^{n} h_i P_{ID_i}, W)\hat{e}(\sum_{i=1}^{n} R_i, T)$. If the above equation holds, output $C = (\{R_i\}_{i=1}^{n}, \{c_i\}_{i=1}^{n}, S)$ and $\{m_i\}_{i=1}^{n}$.

C. Security Model of CASS

Two types of adversaries are generally considered in CPKC. A Type I adversary $\mathcal{A}_I$ can replace any entity's public key but does not know the master secret key. A Type II adversary $\mathcal{A}_{II}$ may access to the master secret key but is not allowed to replace the public key of an entity. As an anonymous aggregate signcryption scheme in CPKC, it has to satisfy data confidentiality, unforgeability and sender anonymity even under the attacks from $\mathcal{A}_I$ and $\mathcal{A}_{II}$.

Definition 2 (Data Confidentiality): A CASS holds data confidentiality if there is no probabilistic polynomial-time (PPT) adversary of Type I or Type II can win Game 1 with a non-negligible advantage.

Game 1: It is played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}_I/\mathcal{A}_{II}$, and has the following four stages. In the first stage, $\mathcal{C}$ initializes the system parameters and the master secret key. The public parameters are passed to $\mathcal{A}_I/\mathcal{A}_{II}$. If the adversary is $\mathcal{A}_{II}$, the master secret key is passed to $\mathcal{A}_{II}$ as well. In the second stage, the $\mathcal{A}_I/\mathcal{A}_{II}$ can ask PPK, PK, SK, Signcrypt, and AggUnSignCrypt queries, where PK and SK are used to model UKG(D_i) in an anonymous CASS. For $\mathcal{A}_I$, he may also replace the public key of an entity by querying RPK. In the third stage, $\mathcal{A}_I/\mathcal{A}_{II}$ outputs $n+1$ distinct challenge identities $\{ID_i^*\}_{i=1}^{n}, ID_j^*$, an aggregate keyword Δ^* and two sets of messages $\mathbb{M}_0^* = \{m_{0,i}^*\}_{i=1}^{n}$, $\mathbb{M}_1^* = \{m_{1,i}^*\}_{i=1}^{n}$ (for $i \in [1, n]$, $|m_{0,i}^*| = |m_{1,i}^*|$). $\mathcal{C}$ randomly chooses a bit μ, and then returns a challenge aggregated ciphertexts C^* on the messages in $\mathbb{M}_\mu^*$ to $\mathcal{A}_I/\mathcal{A}_{II}$. In the final stage, $\mathcal{A}_I/\mathcal{A}_{II}$ can still make the above queries, except the SK query on the ID_j^* and the AggUnSignCrypt query on C^*. Finally, $\mathcal{A}_I/\mathcal{A}_{II}$ outputs a bit μ'. If $\mu' = \mu$, $\mathcal{A}_I/\mathcal{A}_{II}$ wins the game.

Definition 3 (Unforgeability): A CASS is unforgeability if there is no PPT adversary of Type I or Type II can win Game 2 with a non-negligible probability.

Game 2: It is played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}_I/\mathcal{A}_{II}$ and has the following three stages. The first two stages are the same as those stages in Game 1. In the final stage, $\mathcal{A}_I/\mathcal{A}_{II}$ outputs a forged aggregated ciphertext C^* corresponding to $\{ID_i^*\}_{i=1}^{n}, ID_j^*$, $\{PK_i^*\}_{i=1}^{n}, PK_j^*, \Delta^*$ and a set of messages $\mathbb{M}^* = \{M_i^*\}_{i=1}^{n}$. If C^* can pass the verification test, $\mathcal{A}_I/\mathcal{A}_{II}$ wins the game.

Definition 4 (Sender Anonymity): A CASS fulfills sender anonymity if no PPT adversary of Type I or Type II can win Game 3 with a non-negligible probability.

Game 3: It is played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}_I/\mathcal{A}_{II}$ and is the same as Game 1 except the third stage. In the third stage, $\mathcal{A}_I/\mathcal{A}_{II}$ outputs two sender identity sets of n distinct identities $\mathbb{ID}_0^* = \{ID_{0,i}\}_{i=1}^{n}$, $\mathbb{ID}_1^* = \{ID_{1,i}\}_{i=1}^{n}$, a receiver's identity ID_j^*, an aggregate keyword Δ^* and a message set $\mathbb{M}^* = \{m_i^*\}_{i=1}^{n'}$. $\mathcal{C}$ randomly chooses a bit μ, and returns a challenge aggregated ciphertext C^* generated by the $\mathbb{ID}_\mu^*$ on message $\mathbb{M}^*$ to $\mathcal{A}_I/\mathcal{A}_{II}$.

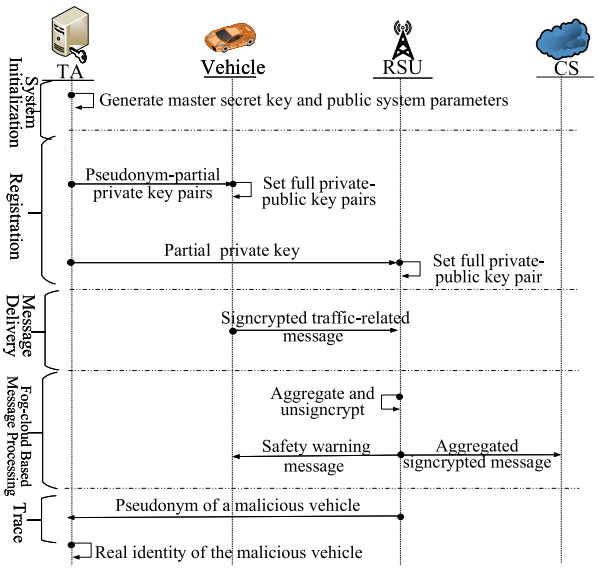


Fig. 2. Overview of PPAAS.

V. The Proposed PPAAS

We present our PPAAS for safety warning system in fog-cloud based VANETs based on our anonymous CASS.

A. High-Level Description

As shown in Fig 2, our PPAAS consists of five stages, namely: *System Initialization*, *Registration*, *Message Delivery*, *Fog-cloud Based Message Processing*, and *Trace*.

In the first stage, the TA generates the master secret key and public system parameters. In the second stage, all the vehicles and the RSUs in the system are enrolled by the TA. That is vehicles and the RSUs generate their full private-public key pairs with the help of the TA. In the third stage, each vehicle delivers signcrypted traffic-related messages to its nearest RSU(s). In the fourth stage, an RSU aggregates and unsigncrypt ciphertexts generated under the same aggregate keyword (e.g., current timestamp). If these messages are verified, the RSU executes a warning precision algorithm (e.g., the algorithm proposed in [27]) and generates a safety warning message if necessary. Then, the RSU broadcasts the safety warning message to the vehicles in its region, it can also transfer this safety warning message to its nearby RSU(s). Finally, the RSU uploads the aggregated ciphertext to the CS for later use. In the stage of trace, the TA can recover the real identity of a malicious vehicle.

We will now describe these stages in detail.

B. System Initialization

In this stage, the TA generates the public system parameters and the master secret key which is used to issue partial private keys for the entities (i.e., vehicles or RSUs). To generate the public system parameters $para'$ and the master secret key (ξ, λ), the TA first runs $Setup(\kappa)$ of our CASS to get $para = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, P_{pub}, H_1 \sim H_6 \rangle$ and λ, and then does the following:

1) Choose a secure symmetric key encryption scheme $E_K(\cdot)/D_K(\cdot)$ and a key ξ, where ξ, $E_K(\cdot)$ and $D_K(\cdot)$ are the key, encryption algorithm and decryption algorithm corresponding to the symmetric key encryption scheme; choose a secure certificateless signature scheme $\Sigma_{sk}(\cdot)/V_{pk}(\cdot)$, where $\Sigma_{sk}(\cdot)/V_{pk}(\cdot)$ is the signature generation/verification algorithm.
2) Set (ξ, λ) as the master secret key and set the system public key as $P_{pub} = \lambda P$.
3) Publish $para' = \langle para, E_K(\cdot)/D_K(\cdot), \Sigma_{sk}(\cdot)/V_{pk}(\cdot)\rangle$.

C. Registration

Each entity (vehicle or RSU) in the system has to be registered with the TA to generate its private-public key pair(s).

For a vehicle, the TA generates a pool of pseudonyms and the corresponding partial private keys for the vehicle at first, since the vehicle has to change its pseudonym to protect its privacy. The full private-public key pairs of the vehicle are generated based on the corresponding partial private keys and used to generate signcrypted traffic-related messages. The detailed process is described in the following *Vehicle Registration* protocol.

Vehicle Registration: Let the vehicle is V_i. The protocol has following steps: 1)

1) V_i first submits registration request $(RID_{V_i}, \tilde{n})$ to the TA through a secure channel, where RID_{V_i} is its real identity and $\tilde{n}$ is the number of pseudonym-partial private key pairs requested.
2) Upon receiving the registration request, the TA sets the pseudonym of V_i as $p_{i,j} = Enc_{\xi}(RID_{V_i} \| j) \| vp_j$ $(1 \leq j \leq \tilde{n})$, where vp_j refers to a valid period. The TA then runs $PPK(\lambda, p_{i,j})$ of our CASS to get partial-private key $D_{i,j}$. Finally, $\mathbb{PD} = \{p_{i,j}, D_{i,j}\}_{j=1}^{\tilde{n}}$ is sent to V_i through the secure channel.
3) After receiving $\mathbb{PD}$, V_i runs $UKG(D_{i,j})$ of our CASS to get the corresponding full private key $(x_{V_i,j}, D_{i,j})$ and public key $P_{V_i,j}$ for $1 \leq j \leq \tilde{n}$.

Unlike a vehicle, we just need to generate a long-time full private-public key pair for an RSU, since we do not need to protect the privacy of the RSU. To generate the key pair for an RSU, the TA first generates a partial private key based on the real identity of the RSU, then the RSU generates its long-time full private-public key pair based on this partial private key. The detailed process is described in the following *RSU Registration* protocol.

RSU Registration: Let the RSU be R_i. The protocol runs as follows:

1) R_i sends its real identity RID_{R_i} to the TA through a secure channel.
2) Upon receiving the identity of R_i, the TA runs $PPK(\lambda, RID_{R_i})$ of our CASS to get the partial-private key D_{R_i} of R_i and sends D_{R_i} to R_i through the secure channel.
3) Once R_i receives D_{R_i}, it runs $UKG(D_{R_i})$ of our CASS to get its long-time full private key (x_{R_i}, D_{R_i}).

D. Message Delivery

In this stage, vehicles generate signcrypted traffic-related messages and deliver them to their nearby RSUs.

In our scheme, each RSU has to periodically broadcast its public key and a signature on the public key signed using $\Sigma_{sk}(\cdot)/V_{pk}(\cdot)$ and a fresh aggregate keyword used to ensure secure signature aggregation in its communication range. When a vehicle enters the communication range of an RSU, it has to verify the validity of the signature on the public key of the RSU. We note that, since the RSUs are connected through a wired network, a vehicle may pre-download all the public key-signature pairs of all RSUs on his driving route and check the validity of the public keys in advance.

Assume a vehicle V_i with current pseudonym $p_{i,j}$, public key $P_{V_i,j}$ and full private key $(x_{V_i,j}, D_{i,j})$ enters R_l's communication range, R_l's public key is P_{R_l} whose validity is already verified, and the current aggregate keyword broadcasted by R_l is Δ. Suppose V_i will send a traffic-related message $m_i = Ts_i||Lon_i||Lat_i||Drec_i||Sd_i||Ac_i$ [27] to R_l, where Ts_i, Lon_i, Lat_i, $Drec_i$, Sd_i and Ac_i represent timestamp, longitude, latitude, direction, speed, and acceleration, respectively. It runs $SignCrypt(m_i, \Delta_t, p_{ij}, RID_{R_l}, P_{V_i}, P_{R_l})$ of our CASS to get the signcrypted traffic-related messages (i.e., ciphertext $C_i = (R_i, c_i, s_i)$) and sends C_i to R_l.

E. Fog-Cloud Based Message Processing

In a high density traffic scenario, an RSU will receive many ciphertexts from surrounding vehicles in a short period of time. If these ciphertexts are unsigncrypted one by one, it will seriously affect the response time of a safety warning system and is not suitable for time critical application. Further, if the RSU transmits these ciphertexts to the CS without compression, it will cause heavy communication and storage overheads. In this stage, signcrypted traffic-related messages under the same aggregate keyword received by a fog node (i.e., an RSU) in a period of time, i.e., a batch period, are unsigncrypted to get the corresponding traffic-related messages and aggregated into an aggregated ciphertext. With the traffic-related messages, the RSU then can generate and broadcast a safety warning message based on a prediction algorithm. The details of fog-cloud based message process come as follows:

Suppose an RSU R_l received n ciphertexts $\{C_i = (R_i, c_i, s_i)\}_{i=1}^{n}$ under the same aggregate keyword Δ from $V_1, \ldots, V_n$ in a batch period, it first runs the $AggUnSignCrypt(\{C_i\}_{i=1}^{n}, \Delta, RID_{R_l}, P_{R_l}, sk_{R_l})$ of our CASS to get the pseudonym of the senders (i.e., $V_1, \ldots, V_n$), the traffic-related messages $\{m_i\}_{i=1}^{n}$ and an aggregated ciphertext $C = (\{R\}_{i=1}^{n}, \{c_i\}_{i=1}^{n}, S)$. Then R_l runs $Pre(\{m_i\}_{i=1}^{n})$ to generate a warning message, where Pre is a prediction algorithm in [27] and broadcasts the warning message to the vehicles in its region if necessary. We note that the RSU can also share these messages with its nearby RSUs, so that they can cooperatively generate a safety warning message (or safety warning messages) with improved profundity and precision. Finally, the RSU forwards the aggregated ciphertext C to the CS.

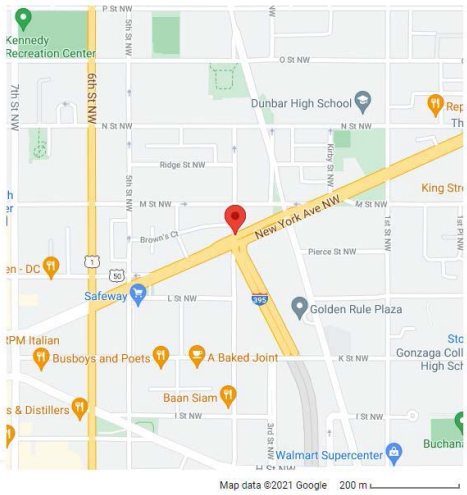


Fig. 3. The Map of The Simulation Scenario.

F. Trace

As mentioned above, a vehicle may be malicious. For instance, a vehicle may send a fake message to an RSU. Therefore, it should be possible to trace the real identity of the malicious vehicle. This stage is used for the TA to recover the real identity of a malicious vehicle.

When an RSU finds a fake message but the corresponding aggregate ciphertext is valid, it will send the pseudonym of the corresponding malicious vehicle to the TA. Assume the pseudonym of the malicious vehicle is p^*_{ij}. The TA just needs to compute $RID^*_{V_i} \| P^*_{V_i} = Dec_\xi(p^*_{i,j}) = Dec_\xi(Enc_\xi(RID^*_{V_i} \| P^*_{V_i}))$ to retrieve the real identity $RID^*_{V_i}$ of the vehicle. It is easy to see that the TA can recover the real identity of a vehicle in constant time. Finally, the malicious vehicle will be punished by the TA.

VI. Simulation and Comparison

Simulations and comparisons are presented to show the efficiency of our PPAAS in this section.

A. Simulation

The simulations were performed with VanetMobiSim, MIRACL cryptographic library and NS-3 on an Ubuntu machine with an Intel Core i7-3700 at a frequency of 3.4 GHz. We implemented a BN curve with 128-bit security level. The road scenario is an area of 1.0×1.0 km^2 shown in Fig. 3. The vehicles were randomly generated with average speed 56 km/h. The communication range of a vehicle was 300 m. The channel bandwidth bound was 6 Mb/s. Vehicles broadcast messages every 100 ms. The ciphertext package size was 212 bytes.

Let $\mathbb{D}$ denote the simulation area, $L_\mathbb{D}$ denote the total number of vehicles in $\mathbb{D}$, τ denote a batch period, $T^i_{w,\tau}$ denote the average waiting time of a message sent by the i-th vehicle before it is processed by an RSU in τ, $T^{N_\tau}_{ver}$ denote the time cost of aggregating and unsigncrypting the ciphertexts received by the RSU in τ, $N_{auth,\tau}$ denote the maximum number of messages received by the RSU in τ that can be verified, $N_{R,\tau}$ denote the total number of messages that received by the RSU. We consider the following performance metrics: D_{msgpro} and R_{loss}. D_{msgpro} refers to the average aggregation and unsigncryption delay. It is the average time delay for a signcrypted message sent by a vehicle to be processed by an RSU, which is defined to be: $D_{msgpro} = \frac{1}{L_\mathbb{D}} \sum_{i=1}^{L_\mathbb{D}} T^i_{w,\tau} + T^{N_\tau}_{ver}$. R_{loss} denotes the aggregation and unsigncryption loss ratio. It is the average probability that a message is received by an RSU but cannot be aggregated and unsigncrypted, which is defined to be $R_{loss} = 1 - \frac{N_{auth,\tau}}{N_{R,\tau}}$. The simulation results are shown in Fig. 4 and Fig. 5.

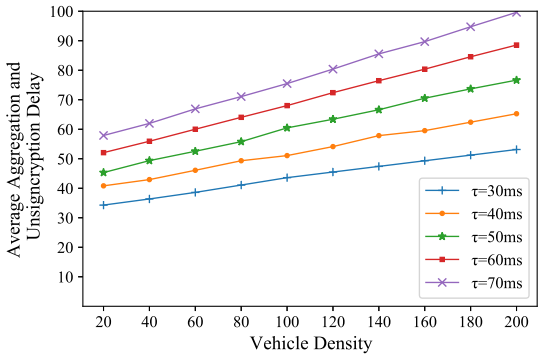


Fig. 4. Average Aggregation and Unsigncryption Delay.

Fig. 4 shows D_{msgpro} for different vehicle density and different batch period τ. It is easy to see that the average message process delay is less than the maximum allowable delay (i.e., 100 ms) [5]. The D_{msgpro} increases with τ for fixed vehicle density. For a fixed τ, the delay is mainly determined by the number of vehicles, and grows slowly as the vehicle number grows. This is due to the fact that the execution time of our *AggUnSignCrypt* algorithm increases slowly with the number of vehicles. In other words, it shows that our PPAAS is suitable for safety warning system that requires low latency. We note that the smaller τ is, the smaller D_{msgpro} is. However, it may increase the authentication loss rate, as discussed below.

Fig. 5 shows R_{loss} for different vehicle density and different batch period τ. It's obvious when $\tau \leq 50$ ms and the number of vehicles is big, the message loss rate is high. This is because the aggregation authentication period is not long enough, so that only a few messages are processed which leaves the advantage of aggregation authentication under-represented. When $\tau > 60$ ms, the message loss rate is almost 0 for any vehicle density. Given the above result, it is an appropriate decision to set $\tau = 60$ ms.

According to ETSI TS 101 539-3 [30], the maximum end-to-end latency for such application is 300 ms. From our simulation, one can observe that the delay (i.e., average aggregation and unsigncryption delay) is less than 100 ms even when the vehicle density is 200. Therefore, our scheme is efficient and practical.

B. Efficiency Comparison

The efficiency of our scheme is mainly dominated by the underlining anonymous CASS. Thus, we compare the

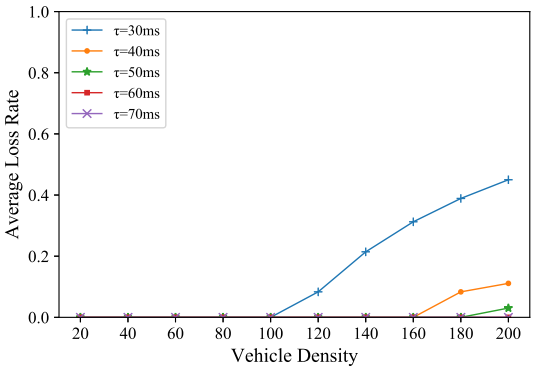


Fig. 5. Average Message Loss Rate.

computation and communication costs of our anonymous CASS with those of the existing provably secure CASSs [8], [9], [32]–[34].

The anonymous CASSs in [8], [9], [32]–[34] are designed from Type 1 bilinear map which is less efficient than Type 2 bilinear map. We note that our scheme can also be implemented using Type 1 bilinear map by simply setting $\mathbb{G}_1 = \mathbb{G}_2$. To be fair, we compare the computation and communication costs of our scheme implemented using Type 1 bilinear map with those of the schemes in [8], [9], [32]–[34]. We note that the efficiency of an anonymous CASSs is dominated by *SignCrypt* and *AggUnSignCrypt* algorithms. Thus, we only compare the efficiency of these two algorithms in different anonymous CASSs.

Table II shows the comparison result regarding to the computation cost, in which $T_{sm_1}/T_{m_T}/T_{bm}/T_H$ denotes the time cost to compute a scale multiplication operation in $\mathbb{G}_1$/a multiplication operation in $\mathbb{G}_T$/a bilinear map operation/a hash to group operation. We note that in the table only costly operations are listed, i.e., multiplication and bilinear map operations, in which a bilinear map operation is much more costly than a multiplication operation. Even when the SS2 curve is chosen (which results in lower computation costs for the above operations, in comparison to commonly used SSP curve), the time costs of the operations, i.e., $T_{sm_1} \approx 1.448$ ms, $T_{m_T} \approx 0.018$ ms, $T_{bm} \approx 11.979$ ms, and $T_h \approx 1.779$ ms, are generally much higher than the time costs of these operations implemented on BN curve which are $T_{sm_1} \approx 0.113$ ms, $T_{m_T} \approx 0.019$ ms, $T_{bm} \approx 3.733$ ms, and $T_h \approx 0.068$ ms. From the table, we also observe that the *SignCrypt* algorithms in [8], [9], [34] and this paper do not involve bilinear map operation, and are more efficient than that those presented in [32], [33]. As for the *AggUnSignCrypt* algorithm, the number of bilinear map operations for the scheme in [8], [32], [33] and [34] is linear in terms of the number of ciphertext received. In contrast, both our scheme and the scheme in [9] only require a constant number of bilinear map operations. Although the *AggUnSignCrypt* algorithm in our scheme has two additional bilinear map operations than that of the scheme in [9], the *AggUnSignCrypt* algorithm in our scheme requires fewer multiplication operations than that of the scheme in [9]. As the number of vehicles nearby an RSU (n) increases,

TABLE II
COMPARISON OF COMPUTATION COST

Scheme	SignCrypt	AggUnSignCrypt
[8], [34]	$7T_{sm_1} + T_H$	$(n+3)T_{bm} + 2nT_{sm_1} + (n+2)T_{m_T} + 2nT_H$
[9]	$6T_{sm_1} + T_H$	$3T_{bm} + 4nT_{sm_1} + T_{m_T} + 2nT_H$
[32]	$4T_{sm_1} + T_{bm} + 3T_H$	$(2n+3)T_{bm} + nT_{sm_1} + (2n+1)T_{m_T} + 2nT_H$
[33]	$3T_{sm_1} + T_{bm} + T_H$	$(2n+3)T_{bm} + nT_{sm_1} + (n+2)T_{m_T} + 2nT_H$
Ours	$6T_{sm_1}$	$5T_{bm} + 3nT_{sm_1} + 3T_{m_T} + 2nT_H$

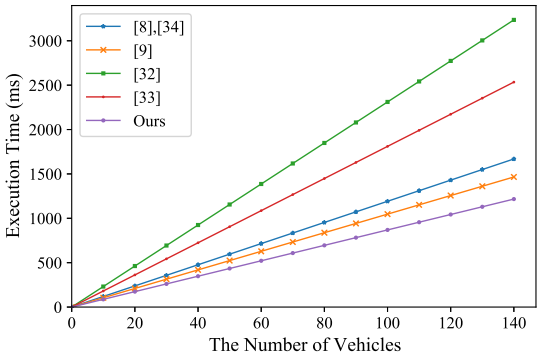


Fig. 6. Computation Cost Comparison of SignCrypt.

the advantage of our *AggUnSignCrypt* algorithm is gradually reflected.

Fig. 6 and Fig. 7 show the simulation results of the *SignCrypt* and *AggUnSignCrypt* algorithms in [8], [9], [32]–[34] and our scheme. From Fig. 6, we can see that the time costs of the *Sincrypt* algorithms in [8], [9], [32]–[34] and our scheme range from 10 ms to 1600 ms, 10 ms to 1500 ms, 20 ms to 3200 ms, 20 ms to 2500 ms and 10 ms to 1200 ms respectively as the number of vehicles ranges from 1 to 140. One can easily observe that the *SignCrypt* algorithm in our scheme is the most efficient. From Fig. 7, we observe that the time costs of the *AggUnSignCrypt* algorithms in [8], [9], [32]–[34] and our scheme range from 50 ms to 2600 ms, 50 ms to 900 ms, 60 ms to 4000 ms and 70 ms to 750 ms respectively as the number of vehicles ranges from 1 to 140. Clearly, the *SignCrypt* algorithm in our scheme is more efficient than those in [8], [32]–[34]. Moreover, when $n > 18$, the *AggUnSignCrypt* algorithm in our scheme becomes more efficient than that of the scheme in [9]. In general, the greater the vehicle density, the greater the probability of a collision. Therefore, our scheme is more suitable than the scheme in [9] for deployment in large scale safety warning systems.

Table III shows the comparison result of the communication cost, where $l_{\mathbb{Z}_q^*}/l_{\mathbb{G}_1}/l_m/l_{ID}$ denotes the bit-length of an element in $\mathbb{Z}_q^*$/an element in $\mathbb{G}_1$/a message m/an identity ID. According to [3], $l_m = 100$ bytes, l_{ID} is at least 16 bytes and $l_{\mathbb{Z}_q^*} = 32$ bytes. We note that even SS2 curve (on which the bit length of an element in $\mathbb{G}_1$ is shorter than that implemented on the commonly used SSP curve) is chosen, the length of an element in $\mathbb{G}_1$, i.e., $l_{\mathbb{G}_1} = 77$ bytes, is much higher than that implemented on BN curve which is $l_{\mathbb{G}_1} = 32$ bytes. From the table, we can see that the length of a single ciphertext in [8], [32]–[34] and our anonymous CASS is 347 bytes while the length of a single ciphertext in [9] is 610 bytes.

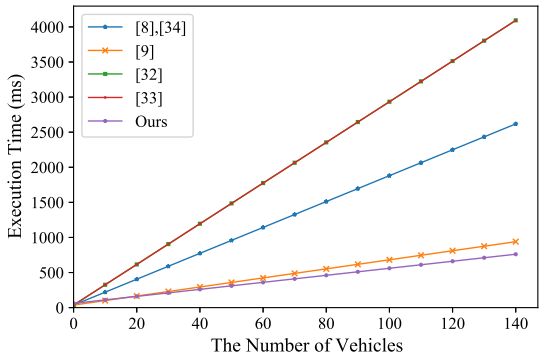


Fig. 7. Computation Cost Comparison of AggUnSignCrypt.

TABLE III
COMPARISON OF COMMUNICATION COST

Scheme	Single ciphertext	Aggregated ciphertext
[8], [34]	$3l_{\mathbb{G}_1} + l_m + l_{ID}$	$(2n+1)l_{\mathbb{G}_1} + n(l_m + l_{ID})$
[9]	$6l_{\mathbb{G}_1} + l_m + l_{ID} + l_{\mathbb{Z}_q^*}$	$(4n+2)l_{\mathbb{G}_1} + n(l_m + l_{ID} + l_{\mathbb{Z}_q^*})$
[32]	$3l_{\mathbb{G}_1} + l_m + l_{ID}$	$(2n+1)l_{\mathbb{G}_1} + n(l_m + l_{ID})$
[33]	$3l_{\mathbb{G}_1} + l_m + l_{ID}$	$(2n+1)l_{\mathbb{G}_1} + n(l_m + l_{ID})$
Ours	$3l_{\mathbb{G}_1} + l_m + l_{ID}$	$(2n+1)l_{\mathbb{G}_1} + n(l_m + l_{ID})$

TABLE IV
SECURITY COMPARISON

Scheme	DC	Auth	ECP	KER
[8]	√	√	×	√
[9]	√	√	×	√
[24]	×	√	×	√
[25]	√	√	×	×
[32]	√	√	×	√
[33]	√	√	×	√
[34]	√	√	×	√
Ours	√	√	√	√

Note: √ denotes that is achieved; × denotes that it is not achieved

The length of an aggregated ciphertext in [8], [32]–[34] and our anonymous CASS is $77 + 270n$ bytes while the length of an aggregated ciphertext in [9] is $154 + 456n$ bytes. Therefore, we can conclude that our scheme is also more suitable for safety warning system than the scheme in [9] in terms of communication overhead. Although the scheme in [8], [32]–[34] and ours have the same communication overhead, our scheme has much lower computation overhead.

C. Security Comparison

We compare the security of our scheme with that of the existing safety warning systems based on V2I communications in [8], [9], [24], [25]. CASSs are also studied in [32]–[34]. We may also apply these CASSs to realize safety warning systems based on V2I communications. Therefore, we also compare the security of our scheme with those based on the CASSs in [32]–[34].

The comparison results are shown in Table IV, in which DC, Auth, ECP, KER denote data confidentiality, authentication, efficiency conditional privacy and key escrow freeness respectively. As shown in Table IV, only our scheme can satisfy all the security requirements.

VII. SECURITY ANALYSIS

In this section, we analyze in detail how our schemes meet the design goals.

Theorem 1: Our anonymous CASS fulfills data confidentiality in the random oracle model under the co-CDH assumption.

In the random oracle model, if there exist a PPT adversary $\mathcal{A}_I/\mathcal{A}_{II}$ who can break data confidentiality of our scheme with non-negligible probability ϵ, asking up to q_i times query to random oracle $H_i (i = 1, \ldots, 6)$, q_{pk} queries on PK, then there is an algorithm $\mathcal{C}$ who can solve the co-CDH problem with advantage $\epsilon' \geq \frac{1}{q_{pk}}\epsilon$.

Proof: $\mathcal{C}$ is given (P, aP, Q, bQ), and his aim is to compute abP with the support of $\mathcal{A}_I/\mathcal{A}_{II}$. $\mathcal{C}$ maintains the following lists: L_i,$(i \in [2, 5])$ for query and response pairs to random oracle H_i, L_K for the user's key pairs record, which are initially empty. Note that, there is no need to model the hash function H_1, H_6 as a random oracle in this case.

First stage: $\mathcal{C}$ first randomly selects $\lambda \in \mathbb{Z}_q^*$ and set it as the msk, computes $P_{pub} = \lambda Q$. Then $\mathcal{C}$ sends the parameters $para = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, P_{pub}, H_1 \sim H_6 \rangle$ to $\mathcal{A}_I/\mathcal{A}_{II}$ and msk is passed to $\mathcal{A}_{II}$ as well.

Second stage: $\mathcal{A}_I/\mathcal{A}_{II}$ can make some queries and $\mathcal{C}$ answers these queries as follows:

H_2 queries: $\mathcal{C}$ keeps a list L_2 of tuples (Δ_i, T_i, β_i). Whenever $\mathcal{A}_I/\mathcal{A}_{II}$ issues a query on $H_2(\Delta_i)$, if the request has been queried before, the corresponding value in L_2 will be returned. Otherwise, $\mathcal{C}$ selects $\beta_i \in_R \mathbb{Z}_q^*$ and sets $T_i = \beta_i Q$. Finally, $\mathcal{C}$ adds (Δ_i, T_i, β_i) to L_2 and returns T_i to $\mathcal{A}_I/\mathcal{A}_{II}$.

H_3 queries: $\mathcal{C}$ keeps a list L_3 of tuples $(\Delta_i, V_i, \gamma_i)$. Whenever $\mathcal{A}_I/\mathcal{A}_{II}$ issues a query on $H_3(\Delta_i)$, if the request has been queried before, the corresponding value in L_3 will be returned. Otherwise, $\mathcal{C}$ selects $\gamma_i \in_R \mathbb{Z}_q^*$ and sets $V_i = \gamma_i Q$. Finally, $\mathcal{C}$ adds $(\Delta_i, V_i, \gamma_i)$ to L_3 and returns V_i to $\mathcal{A}_I/\mathcal{A}_{II}$.

H_4 queries: $\mathcal{C}$ keeps a list L_4 of tuples (Δ_i, W_i, π_i). Whenever $\mathcal{A}_I/\mathcal{A}_{II}$ issues a query on $H_4(\pi_i)$, if the request has been queried before, the corresponding value in L_2 will be returned. Otherwise, $\mathcal{C}$ selects $\pi_i \in_R \mathbb{Z}_q^*$ and sets $W_i = \pi_i Q$. Finally, $\mathcal{C}$ adds (Δ_i, W_i, π_i) to L_4 and returns W_i to $\mathcal{A}_I/\mathcal{A}_{II}$.

H_5 queries: $\mathcal{C}$ keeps an initially empty list L_5 of tuple (U_i, R_i, P_j, K_i). Whenever $\mathcal{A}_I/\mathcal{A}_{II}$ issues a query on $H_5(U_i, R_i, P_j)$, $\mathcal{C}$ does as follows: 1) Check if $\hat{e}(aP, d_j bP) = \hat{e}(P, U_i)$ for all $j = 1, \ldots, n$. If there is a value J, which is satisfies this equation, $\mathcal{C}$ outputs $d_j^{-1} U_i$ and stops. 2) Search the list L_5 for a tuple (U_i, K_i) for some vale of K_i. If such a tuple exists, $\mathcal{C}$ returns K_i to $\mathcal{A}_I/\mathcal{A}_{II}$. Otherwise, if $\hat{e}(R_i, P_j) = \hat{e}(P, U_i)$, select $K_i \in_R \{0, 1\}^k$, return K_i as answer and add (U_i, R_i, P_j, K_i) to L_5.

PK *queries*: $\mathcal{C}$ keeps a list L_K of tuples (ID_i, x_i, P_i), chooses $I \in_R [1, q_{pk}]$. On receiving a query $PK(ID_i)$, $\mathcal{C}$ returns the same answer, if the request has been queried before. Otherwise, $\mathcal{C}$ selects $x_i \in_R \mathbb{Z}_q^*$ and does as follows: If $i = I$, compute $P_i = x_i aP$, add (ID_i, x_i, P_i) to L_K and return P_i to $\mathcal{A}_I/\mathcal{A}_{II}$. Else, compute $P_i = x_i P$, add (ID_i, x_i, P_i) to L_K and return P_i to $\mathcal{A}_I/\mathcal{A}_{II}$.

PPK *queries*: Given a query on $PPK(ID_i)$, Since C knows the msk, it can generate partial private key in the usual manner. Note that, only $\mathcal{A}_I$ may issue this type of query, since the $\mathcal{A}_{II}$ can generate partial private key with the master private key by itself.

SK *queries*: Given a query on $SK(ID_i)$ from $\mathcal{A}_I/\mathcal{A}_{II}$, $\mathcal{C}$ first make $PK(ID_i)$ query then finds the tuple (ID_i, x_i, P_i) in L_K. If $ID_i = ID_I$, $\mathcal{C}$ aborts; otherwise, returns x_i.

RPK *queries*: $\mathcal{A}_I$ can choose any valid public key for a user with identity ID_i. On receiving a query $RPK(ID_i, P_i')$, $\mathcal{C}$ first finds the tuple (ID_i, x_i, P_i) in L_K, if there is no such tuple, $\mathcal{C}$ first makes a $PK(ID_i)$ query and updates (ID_i, x_i, P_i) to $(ID_i, \perp, P_i')$. Note that, $\mathcal{A}_I$ cannot issue this query on any receiver (since the public key of RSU will not be able to be replaced by anyone in our PPAAS) and $\mathcal{A}_{II}$ cannot make this type of query.

SignCrypt *queries*: Given a query on $SignCrypt(\Delta_i, m_i, ID_i, P_i, ID_j, P_j)$ from $\mathcal{A}_I/\mathcal{A}_{II}$, $\mathcal{C}$ first makes $H_2(\Delta_i)$, $H_3(\Delta_i)$, $H_4(\Delta_i)$ if they have not been queried before, then recovers (Δ_i, T_i, β_i) from L_2, $(\Delta_i, V_i, \gamma_i)$ from L_3, (Δ_i, W_i, π_i) from L_4, and generates ciphertext as follows: If $ID_i \neq ID_I$ and the public key of ID_i haven't been replaced by $\mathcal{A}_I$, $\mathcal{C}$ generates C_i as per the $SignCrypt$ algorithm using the sender's private key and receiver's public key. If $ID_i \neq ID_I$ but the public key of ID_i have been replaced by $\mathcal{A}_I$ or $ID_i = ID_I$ (hence, $ID_j \neq ID_I$), $\mathcal{C}$ generates C_i as follows: choose $r_i \in_R \mathbb{Z}_q^*$, compute $R_i = r_i P$, $U_i = r_i P_j$, make a $H_5(U_i, R_i, P_j)$ query to obtain K_i and set $c_i = K_i \oplus (m_i \| ID_i \| P_{ID_i})$, compute $s_i = \lambda H_1(ID_i, 0) + \gamma_i P_i + h_i \lambda H_1(ID_i, 1) + h_i \pi_i P_i + r_i T_i$, where $h_i = H_6(P_i \| P_j \| ID_j \| R_i \| \Delta_i \| m_i)$. Finally, return $C_i = (R_i, c_i, s_i)$ to $\mathcal{A}_I/\mathcal{A}_{II}$.

AggUnsignCrypt *queries*: On receiving a query on $AggUnSignCrypt(\{R_i\}_{i=1}^n, \{c_i\}_{i=1}^n, S, ID_j, \Delta)$ from $\mathcal{A}_I/\mathcal{A}_{II}$, $\mathcal{C}$ makes $H_2(\Delta_i)$, $H_3(\Delta_i)$,$H_4(\Delta_i)$ queries if they have not been queried before, then recovers (Δ_i, T_i, β_i) from L_2, $(\Delta_i, V_i, \gamma_i)$ from L_3, (Δ_i, W_i, π_i) from H_4, (ID_j, x_j, P_j) from L_K, and for $i \in [1, n]$, does the following: If $ID_j \neq ID_I$, does as per the $AggUnSignCrypt$ algorithm. Otherwise, dose as follows: search L_5 to look for a tuple (U_i, R_i, ID_j, K_i), for different values of U_i, such that $\hat{e}(R_i, P_j) = \hat{e}(P, U_i)$. If such an entry exists, the correct value of U_i is found (since $\mathcal{C}$ can not compute U_i), retrieve m_i using the corresponding K_i. Otherwise, place $(*, R_i, ID_j, K_i)$ with a random value $K_i \in_R \{0,1\}^k$ on list L_5, retrieve m_i using this K_i and return $(C, \{m_i\}_{i=1}^n)$.

Third stage: At the end of the second stage, $\mathcal{A}_I/\mathcal{A}_{II}$ outputs two message sets $\mathbb{M}_0^* = \{m_{0,i}^*\}_{i=1}^n$ and $\mathbb{M}_1^* = \{m_{1,i}^*\}_{i=1}^n$, an aggregate keyword Δ^* and $n+1$ identities $\{ID_i^*\}_{i=1}^n$ and ID_R^*. If $ID_R^* \neq ID_I$, $\mathcal{C}$ fails. Otherwise, it constructs a challenge ciphertext as follows: obtain $\{(ID_i^*, x_i^*, P_i^*)\}_{i=1}^n$ from L_K, set $R_i^* = d_i \psi(bQ)$ with random $\{d_i \in_R \mathbb{Z}_q^*\}_{i=1}^n$, select a random bit μ, then obtains hash value $\{K_i\}_{i=1}^n$ from H_5 oracle, $(\Delta^*, T^*, \alpha^*)$ from H_2 oracle, (Δ^*, V^*, β^*) from H_3 oracle, and (Δ^*, W^*, π^*) from H_4 oracle. The component of c_i^* and s_i^* are set to be $c_i^* = K_i \oplus (m_{\mu,i}^* \| ID_i^* \| P_{ID_i^*})$ and $s_i^* = \lambda H_1(ID_i^*, 0) + x_i^* \psi(V^*) + h_i(\lambda H_1(ID_i^*, 1) + x_i^* \psi(W^*)) + \alpha_i^* R_i^*$ respectively, where $h_i = H_6(P_i^* \| P_j^* \| ID_j^* \| R_i^* \| \Delta_i^* \| m_{\mu,i}^*)$. Finally, return $(\{R_i^*\}_{i=1}^n, \{c_i^*\}_{i=1}^n, S^* = \sum_{i=1}^n s_i^*)$ to $\mathcal{A}_I/\mathcal{A}_{II}$.

Final stage: $\mathcal{A}_I/\mathcal{A}_{II}$ can perform new queries, which are treated in the same manner as the second stage. At the end of the simulation, $\mathcal{A}_I/\mathcal{A}_{II}$ will output μ'.

Now we compute the probability of $\mathcal{C}$ solving the given instance of co-CDH problem. When analyzing the advantage of $\mathcal{C}$, we consider the following four events: $E1$: $\mathcal{A}_I/\mathcal{A}_{II}$ choose ID_I as the receiver's identity to be challenged, that is $ID_j^* = ID_I$. $E2$: $\mathcal{C}$ does not abort as a result of any SK query of $\mathcal{A}_I/\mathcal{A}_{II}$. $E3$: $\mathcal{A}_I/\mathcal{A}_{II}$ win the game, that is $\mu' = \mu$. $E4$: $\mathcal{A}_I/\mathcal{A}_{II}$ performed a challenge related query on H_5 oracle. $\mathcal{C}$ succeeds if all of the above four events happen. Obviously, the probability of $\mathcal{C}$'s success is $\epsilon' = Pr[E1 \wedge E2 \wedge E3 \wedge E4] \geq \frac{1}{q_{pk}}\epsilon$.

Theorem 2: Our anonymous CASS fulfills unforgeability in the random oracle model under the co-CDH assumption.

In the random oracle model, if there is a PPT adversary $\mathcal{A}_I/\mathcal{A}_{II}$ who can forge an aggregated signcrypt ciphertext of our CASS with a non-negligible probability ϵ, asking up to q_i times query to random oracle $H_i (i = 1, \ldots, 6)$ q_k queries on PPK, q_s queries on SignCrypt. Then there is an algorithm $\mathcal{C}$ who can solve the co-CDH problem with advantage $\epsilon' \geq (1-\frac{1}{q_1})^{q_{pk}}(1-\frac{1}{max(q_{pk},q_1)}\frac{1}{q_2}(1-\frac{1}{q_6'}))^{q_s} \times \frac{1}{max(q_{pk},q_1)}\frac{1}{q_2}(1-\frac{1}{q_6'})\epsilon$

Proof: $\mathcal{C}$ is given (P, aP, Q, bQ), and his aim is to compute abP with the support of $\mathcal{A}_I/\mathcal{A}_{II}$. $\mathcal{C}$ maintains the following lists: L_i $(i = 1, \ldots, 6)$ for query and response pairs to random oracle H_i, L_K for the user's key pairs record, which are initially empty. $\mathcal{C}$ first chooses a bit $b \in_R \{0,1\}$. If $b = 0$, $\mathcal{C}$ plays the game according to the strategy for $\mathcal{A}_I$, otherwise plays according to the strategy for $\mathcal{A}_{II}$. Strategies for different adversaries are described in detail below.

First stage: As for $\mathcal{A}_{II}$, $\mathcal{C}$ sets $P_{pub} = bQ$, $para = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, P_{pub}, H_1 \sim H_6 \rangle$ and sends $para$ to the $\mathcal{A}_I$. As for $\mathcal{A}_{II}$, $\mathcal{C}$ selects $\lambda \in \mathbb{Z}_q^*$ as msk, sets $P_{pub} = \lambda Q$, and sends $para = \langle \mathbb{G}_1, \mathbb{G}_2, \hat{e}, P, P_{pub}, H_1 \sim H_6 \rangle$ and msk to $\mathcal{A}_{II}$.

Second stage: $\mathcal{A}_I/\mathcal{A}_{II}$ can make some queries and $\mathcal{C}$ answers these queries as follows:

H_1 *queries*: $\mathcal{C}$ keeps a list L_1 of tuples $(ID_i, \alpha_{i,0}, \alpha_{i,0}', \alpha_{i,1}, \alpha_{i,1}', Q_{i,0}, Q_{i,1})$. $\mathcal{C}$ picks $I \in_R [1, q_1]$. Whenever $\mathcal{C}$ receives a query $H_1(ID_i, j)$, $(j \in \{0,1\})$ from $\mathcal{A}_I$, $\mathcal{C}$ does the following: 1) If there is a tuple $(ID_k, \alpha_{k,0}, \alpha_{k,0}', \alpha_{k,1}, \alpha_{k,1}', Q_{k,0}, Q_{k,1})$ in L_1 such that $ID_i = ID_k$, return $Q_{k,j}$. 2) Else if $i = I$, choose $\alpha_{i,0}, \alpha_{i,0}', \alpha_{i,1}, \alpha_{i,1}' \in_R \mathbb{Z}_q^*$, compute $Q_{i,0} = \alpha_{i,0} P + \alpha_{i,0}' aP$, $Q_{i,0} = \alpha_{i,1} P + \alpha_{i,1}' aP$, add $(ID_i, \alpha_{i,0}, \alpha_{i,0}', \alpha_{i,1}, \alpha_{i,1}', Q_{i,0}, Q_{i,1})$ to L_1 and return $Q_{i,j}$. 3) Else randomly choose $\alpha_{i,0}, \alpha_{i,1} \in_R \mathbb{Z}_q^*$, compute $\alpha_{i,0}' = 0$, $\alpha_{i,1}' = 0$, $Q_{i,0} = \alpha_{i,0} P$, $Q_{i,1} = \alpha_{i,1} P$, add $(ID_i, \alpha_{i,0}, \alpha_{i,0}', \alpha_{i,1}, \alpha_{i,1}', Q_{i,0}, Q_{i,1})$ to L_1 and return $Q_{i,j}$.

Note that when consider $\mathcal{A}_{II}$, we need not model H_1 as a random oracle.

H_2 *queries*: $\mathcal{C}$ keeps a list L_2 of tuples (Δ_i, T_i, β_i), picks $J \in_R [1, q_2]$. Whenever $\mathcal{A}_I/\mathcal{A}_{II}$ issues a query $H_2(\Delta_i)$, if the request has been queried before, the corresponding value in L_2 will be returned.. Otherwise, $\mathcal{C}$ selects $\beta_i \in_R \mathbb{Z}_q^*$, if $i = J$,

computes $T_i = \beta_i Q$, else computes $T_i = \beta_i bQ$. Finally, $\mathcal{C}$ adds (Δ_i, T_i, β_i) to L_2 and returns T_i to $\mathcal{A}_I/\mathcal{A}_{II}$.

H_3 *queries*: $\mathcal{C}$ keeps a list L_3 of tuples $(\Delta_i, V_i, \gamma_i, \gamma_i')$. Whenever $\mathcal{A}_I/\mathcal{A}_{II}$ issues a query $H_3(\Delta_i)$, if the request has been queried before, the corresponding value in L_3 will be returned. Otherwise, $\mathcal{C}$ randomly selects $\gamma_i \in_R \mathbb{Z}_q^*/\gamma_i, \gamma_i' \in_R \mathbb{Z}_q^*$ and computes $V_i = \gamma_i Q/V_i = \gamma_i Q + \gamma_i' bQ$. Finally, $\mathcal{C}$ adds $(\Delta_i, V_i, \gamma_i, \perp)/(\Delta_i, V_i, \gamma_i, \gamma_i')$ to L_3 and returns V_i.

H_4 *queries*: $\mathcal{C}$ keeps a list L_4 of tuples $(\Delta_i, W_i, \pi_i, \pi_i')$. Whenever $\mathcal{A}_I/\mathcal{A}_{II}$ issues a query $H_4(\Delta_i)$, if the request has been queried before, the corresponding value in L_3 will be returned. Otherwise, $\mathcal{C}$ randomly selects $\pi_i \in_R \mathbb{Z}_q^*/\pi_i, \pi_i' \in_R \mathbb{Z}_q^*$ and computes $W_i = \pi_i Q/W_i = \pi_i Q + \pi_i' bQ$. Finally, $\mathcal{C}$ adds $(\Delta_i, W_i, \pi_i, \perp)/(\Delta_i, W_i, \pi_i, \pi_i')$ to L_4 and returns W_i.

H_5 *queries*: $\mathcal{C}$ keeps a list L_5 of tuples (U_i, R_i, PK, K_i). Whenever $\mathcal{A}_I/\mathcal{A}_{II}$ issues a query $H_5(U_i, R_i, PK)$, $\mathcal{C}$ returns the same answer from L_5 if the request has been queried before. Otherwise, $\mathcal{C}$ randomly selects $K_i \in_R \{0,1\}^k$, returns K_i to $\mathcal{A}_I/\mathcal{A}_{II}$ and adds (U_i, R_i, PK, K_i) to L_5.

H_6 *queries*: $\mathcal{C}$ keeps a list L_6 of tuples $(P_i, P_j, ID_j, R_i, \Delta_i, m_i, h_i)$. Whenever $\mathcal{A}_I/\mathcal{A}_{II}$ issues a query $H_6(P_i, P_j, ID_j, R_i, \Delta_i, m_i)$, $\mathcal{C}$ gives the same answer from L_6 if the request have been queried before. Otherwise, $\mathcal{C}$ dose as follows:

- For a query of $\mathcal{A}_I$, $\mathcal{C}$ makes $H_1(ID_i, 0)$ query, finds $(ID_i, \alpha_{i,0}, \alpha_{i,0}', \alpha_{i,1}, \alpha_{i,1}', Q_{i,0}, Q_{i,1})$ from L_1 and does as follows: 1) If $ID_i = ID_I$ and $\Delta_i = \Delta_J$, (here, we assume that $\mathcal{A}_I$ can ask at most $q_6' \leq q_6$ times such kind of queries) choose $K \in [1, q_6']$. If this is the K-th query, set $h_i = -\alpha_{i,0}'/\alpha_{i,1}'$, add $(P_i, P_j, ID_j, R_i, \Delta_i, m_i, h_i)$ to L_6 and return h_i; otherwise, random select $h_i \in_R \mathbb{Z}_q^*$, add $(P_i, P_j, ID_j, R_i, \Delta_i, m_i, h_i)$ to L_6 and return h_i. 2) Else, select $h_i \in_R \mathbb{Z}_q^*$, add $(P_i, P_j, ID_j, R_i, \Delta_i, m_i, h_i)$ to L_6 and return h_i.
- For a query of $\mathcal{A}_{II}$, $\mathcal{C}$ makes $H_3(\Delta_i)$, $H_4(\Delta_i)$, $PK(ID_i)$ queries, finds $(\Delta_i, V_i, \gamma_i, \gamma_i')$ from L_3, $(\Delta_i, W_i, \pi_i, \pi_i')$ from L_4, (ID_i, x_i, P_i) from L_K and does the following: 1) If $\Delta_i = \Delta_J, P_i = P_M$ (here, we assume that $\mathcal{A}_{II}$ can ask at most $q_6' \leq q_6$ times such kind of queries) choose $K \in [1, q_6']$. If this is the K-th query, set $h_i = -\gamma_i'/\pi_i'$, add $(P_i, P_j, ID_j, R_i, \Delta_i, m_i, h_i)$ to L_6 and return h_i; otherwise, select $h_i \in_R \mathbb{Z}_q^*$, add $(P_i, P_j, ID_j, R_i, \Delta_i, m_i, h_i)$ to L_6 and return h_i. 2) Else, randomly select a $h_i \in_R \mathbb{Z}_q^*$, add $(P_i, P_j, ID_j, R_i, \Delta_i, m_i, h_i)$ to L_6 and return h_i.

PPK *queries*: $\mathcal{C}$ keeps a list L_K of tuple $(ID_i, x_i, D_{i,0}, D_{i,1}, P_i)$. Note that only $\mathcal{A}_I$ may issue this type of query, since $\mathcal{A}_{II}$ can generate partial private key by itself. When $\mathcal{A}_I$ makes $PPK(ID_i)$, $\mathcal{C}$ returns the same answer from L_K if the request have been queried before. Otherwise, $\mathcal{C}$ checks if there is a tuple $(ID_i, \alpha_{i,0}, \alpha_{i,0}', \alpha_{i,1}, \alpha_{i,1}', Q_{i,0}, Q_{i,1})$ in L_1, if no, $\mathcal{C}$ makes an H_1 query on $ID_{i,j}$ for $j = 0$ or 1, finally does as follows: 1) If $ID_i = ID_I$, abort. 2) Else if there is a tuple $(ID_i, x_i, D_{i,0}, D_{i,1}, P_i)$ in L_K, set $D_{i,0} = \alpha_{i,0}\psi(P_{pub})$, $D_{i,1} = \alpha_{i,1}\psi(P_{pub})$ and return $(D_{i,0}, D_{i,1})$. 3) Else, set $D_{i,0} = \alpha_{i,0}\psi(P_{pub})$, $D_{i,1} = \alpha_{i,1}\psi(P_{pub})$, $x_i = P_i = \perp$, add $(ID_i, x_i, D_{i,0}, D_{i,1}, P_i)$ to L_K and return $D_{i,0}, D_{i,1}$.

PK *queries*: Given a query $PK(ID_i)$, $\mathcal{C}$ returns the same answer from L_K if the request has been queried before. Otherwise, $\mathcal{C}$ does as follows: 1) For a query of $\mathcal{A}_I$, if there's a tuple $(ID_i, x_i, D_{i,0}, D_{i,1}, P_i)$ in L_K and the public key P_i of ID_i is $\perp$, choose $x_i' \in_R \mathbb{Z}_q^*$, compute $P_i' = x_i' P$, return P_i' and update $(ID_i, x_i, D_{i,0}, D_{i,1}, P_i)$ to $(ID_i, x_i', D_{i,0}, D_{i,1}, P_i')$. Otherwise, choose $x_i \in_R \mathbb{Z}_q^*$, compute $P_i = x_i P$, return P_i, set $D_{i,0} = D_{i,1} = \perp$ and add $(ID_i, x_i, D_{i,0}, D_{i,1}, P_i)$ to L_K. 2) For a query of $\mathcal{A}_{II}$, $\mathcal{C}$ keeps a list L_K of tuples (ID_i, x_i, P_i), chooses $M \in_R [1, q_{pk}]$, selects $x_i \in_R \mathbb{Z}_q^*$ and does as follows: If $i = M$, compute $P_i = x_i aP$, add (ID_i, x_i, P_i) to L_K and return P_i. Otherwise, set $P_i = x_i P$, add (ID_i, x_i, P_i) to L_K and return P_i.

SK *queries*: Given a query $SK(ID_i)$, $\mathcal{C}$ returns the same answer from L_K if the request has been queried before. Otherwise, $\mathcal{C}$ makes $PK(ID_i)$ query, finds $(ID_i, x_i, D_{i,0}, D_{i,1}, P_i)/(ID_i, x_i, P_i)$ in L_K and return x_i to $\mathcal{A}_I/\mathcal{A}_{II}$. Note that when answering queries of $\mathcal{A}_I$, x_i may be $\perp$ and $\mathcal{C}$ will abort when $\mathcal{A}_{II}$ makes $SK(ID_M)$.

RPK *queries*: $\mathcal{A}_I$ can choose any valid public key for a user with identity ID_i. Given a query $RPK(ID_i, P_i')$, $\mathcal{C}$ finds $(ID_i, x_i, D_{i,0}, D_{i,1}, P_i)$ from L_K, if there's no such tuple, $\mathcal{C}$ makes a $PK(ID_i)$ query, then updates $(ID_i, x_i, D_{i,0}, D_{i,1}, P_i)$ to $(ID_i, \perp, D_{i,0}, D_{i,1}, P_i')$. Note that, $\mathcal{A}_{II}$ can't make this type of query.

SignCrypt *queries*: Given a query $SignCrypt(\Delta_i, m_i, ID_i, P_i, ID_j, P_j)$, $\mathcal{C}$ makes $H_2(\Delta_i)$, $H_3(\Delta_i)$, $H_4(\Delta_i)$ queries if they haven't been queried before, then recovers (Δ_i, T_i, β_i) from L_2, $(\Delta_i, V_i, \gamma_i, \perp/\gamma_i')$ from L_3, $(\Delta_i, W_i, \pi_i, \perp/\pi_i')$ from L_4. Note that for a query of $\mathcal{A}_I$, $\mathcal{C}$ also makes $H_1(ID_i, 0)$ and $H_1(ID_i, 1)$ queries if they haven't been queried before, and recovers $(ID_i, \alpha_{i,0}, \alpha_{i,0}', \alpha_{i,1}, \alpha_{i,1}', Q_{i,0}, Q_{i,1})$ from L_1, then does as follows:

- For a query of $\mathcal{A}_I$, $\mathcal{C}$ does the following: a) If $ID_i = ID_I, \Delta_i = \Delta_J$, and $h_i = -\alpha_{i,0}'/\alpha_{i,1}'$, choose $r_i \in_R \mathbb{Z}_q^*$, compute $U_i = r_i P_j$, make $H_5(U_i, R_i, P_j)$ query to obtain K_i, compute $c_i = K_i \oplus (m_i \| ID_i \| P_{ID_i})$ and recover $(P_i, P_j, ID_j, R_i, \Delta_i, m_i, h_i)$ from L_6. Finally, compute $s_i = \alpha_{i,0}\psi(P_{pub}) + \gamma_i P_i + \alpha_{i,1} h_i \psi(P_{pub}) + \pi_i h_i P_i + r_i \psi(T_i)$, and return $C_i = (R_i, c_i, s_i)$. b) Else if $ID_i = ID_I, \Delta_i = \Delta_J$, abort. c) Else if $ID_i = ID_I$, choose $r_i \in_R \mathbb{Z}_q^*$, set $R_i = r_i P - \beta_i^{-1}(Q_{i,0} + h_i Q_{i,1})$. Then recover $(ID_j, x_j, D_{j,0}, D_{j,1}, P_j)$ from L_K, set $U_i = x_j R_i$, make $H_5(U_i, R_i, P_j)$ query to get K_i, compute $c_i = K_i \oplus (m_i \| ID_i \| P_{ID_i})$, recover $(P_i, P_j, ID_j, R_i, \Delta_i, m_i, h_i)$ from L_6, compute $s_i = \gamma_i P_i + \pi_i h_i P_i + r_i \psi(T_i)$ and return $C_i = (R_i, c_i, s_i)$. d) Else, select $r_i \in_R \mathbb{Z}_q^*$, compute $U_i = r_i P_j$ and make a $H_5(U_i)$ query to obtain K_i, then compute $c_i = K_i \oplus (m_i \| ID_i \| P_{ID_i})$ and recover $(P_i, P_j, ID_j, R_i, \Delta_i, m_i, h_i)$ from L_6, set $s_i = \alpha_{i,0}\psi(P_{pub}) + \gamma_i P_i + \alpha_{i,1} h_i \psi(P_{pub}) + \pi_i h_i P_i + r_i \psi(T_i)$ and return $C_i = (R_i, c_i, s_i)$.
- For a query of $\mathcal{A}_{II}$, $\mathcal{C}$ does following: a) If $\Delta_i = \Delta_J, P_i = P_M$, and $h_i = -\gamma_i'/\pi_i'$, select $r_i \in_R \mathbb{Z}_q^*$, compute $R_i = r_i P$, $U_i = r_i P_j$, make a $H_5(U_i, R_i, P_j)$ query to obtain K_i, set $c_i = K_i \oplus (m_i \| ID_i \| P_{ID_i})$, recover $(P_i, P_j, ID_j, R_i, \Delta_i, m_i, h_i)$ from L_6, set $s_i = \lambda(H_1(ID_i, 0) + h_i H_1(ID_i, 1)) + \gamma_i P_i + \pi_i h_i P_i + r_i \psi(T_i)$

and return $C_i = (R_i, c_i, s_i)$. b) Else if $\Delta_i = \Delta_J$, $P_i = P_M$, abort. c) Else if $P_i = P_M$, choose $r_i \in_R \mathbb{Z}_q^*$, set $R_i = r_i P - \beta_i^{-1}(\gamma_i' + \pi_i' h_i) P_i$, recover $(ID_j, x_j, D_{j,0}, D_{j,1}, P_j)$ from L_K, compute $U_i = x_j R_i$, make a $H_5(U_i, R_i, P_j)$ query to obtain K_i, set $c_i = K_i \oplus (m_i \| ID_i \| P_{ID_i})$, recover $(P_i, P_j, ID_j, R_i, \Delta_i, m_i, h_i)$ from L_6, set $s_i = \lambda(H_1(ID_i, 0) + h_i H_1(ID_i, 1)) + (\gamma_i + \pi_i h_i) P_i + r_i \psi(T_i)$ and return $C_i = (R_i, c_i, s_i)$. d) Else, select $r_i \in_R \mathbb{Z}_q^*$, compute $R_i = r_i P$, $U_i = r_i P_j$ and make a $H_5(U_i, R_i, P_j)$ query to obtain K_i, compute $c_i = K_i \oplus (m_i \| ID_i \| P_{ID_i})$ and recover $(P_i, P_j, ID_j, R_i, \Delta_i, m_i, h_i)$ from L_6, set $s_i = \lambda(H_1(ID_i, 0) + h_i H_1(ID_i, 1)) + x_i \psi(V_i) + x_i h_i \psi(W_i) + r_i \psi(T_i)$ and return $C_i = (R_i, c_i, s_i)$.

AggUnSignCrypt *queries*: On receiving a query $AggUnSignCrypt(\{R_i\}_{i=1}^n, \{c_i\}_{i=1}^n, S, ID_j, \Delta)$ from $\mathcal{A}_I$ /$\mathcal{A}_{II}$, $\mathcal{C}$ makes $H_2(\Delta_i)$, $H_3(\Delta_i)$, $H_4(\Delta_i)$ queries if they have not been queried before, then recovers (Δ_i, T_i, β_i) from L_2, $(\Delta_i, V_i, \gamma_i, \perp/\gamma_i')$ from L_3, $(\Delta_i, W_i, \pi_i, \perp/\pi_i')$ from H_4, $(ID_j, x_j, D_{j,0}, D_{j,1}, P_j)/(ID_j, x_j, P_j)$ from L_K, and for $i \in [1, n]$, does the following: compute $U_i = x_j R_i$ and search (U_i, R_i, ID_j, K_i) in L_5. If such entry exists, compute $m_i \| ID_i \| P_{ID_i} = K_i \oplus c_i$, obtain $(P_i, P_j, ID_j, R_i, \Delta_i, m_i, h_i)$ from L_6. Check whether the verify equation holds, if so, return m_i. Otherwise, return $\perp$. Otherwise, select $K_i \in_R \{0, 1\}^k$, retrieve m_i and verify using these values, add (U_i, R_i, ID_j, K_i) to L_5 and return $(C = (\{R_i\}_{i=1}^n, \{c_i\}_{i=1}^n, S_i), \{m_i\}_{i=1}^n)$.

Third stage: Eventually, $\mathcal{A}_I/\mathcal{A}_{II}$ outputs a set of $n+1$ users' identity $\{ID_i^*\}_{i=1}^n, ID_j^*$, n messages $\{m_i^*\}_{i=1}^n$, an aggregate keyword Δ^* and a forged aggregated ciphertext $C^* = (\{R_i^*\}_{i=1}^n, \{c_i^*\}_{i=1}^n, S^*)$.

In order to compute abP, $\mathcal{C}$ recovers (Δ^*, T^*, β^*) from L_2, $(\Delta^*, V^*, \gamma^*, \perp/\gamma'^*)$ from L_3, $(\Delta^*, W^*, \pi^*, \perp/\pi'^*)$ from L_4, and for all $i, i = [1, n]$ recovers h_i from L_6, (ID_i, x_i, P_i) from L_K. For $\mathcal{A}_I$, $\mathcal{C}$ should also recover $(ID_i, \alpha_{i,0}, \alpha_{i,0}', \alpha_{i,1}, \alpha_{i,1}', Q_{i,0}, Q_{i,1})$ from L_1, for all i, $i = [1, n]$. Moreover, it's required that $\Delta^* = \Delta_J$ and there exists $i \in [1, n]$ such that $ID_i^* = ID_I$, $h_i^* \neq -\alpha_{i,0}'/\alpha_{i,1}'$ and $SignCrypt(\Delta^*, m_i^*, ID_j^*, *)$ has never been queried. As for $\mathcal{A}_{II}$, it's required that $\Delta^* = \Delta_J$ and there exists $i \in [1, n]$ such that $P_i^* = P_M$, $h_i^* \neq -\gamma_i'^*/\pi_i'^*$ and $SignCrypt(\Delta^*, m_i^*, ID_i^*, *)$ has never been queried. Without loss of generality, we let $i = 1$. The forged ciphertext must satisfy $\hat{e}(S^*, Q) = \hat{e}(\sum_{i=1}^n Q_{i,0}^* + \sum_{i=1}^n h_i^* Q_{i,1}^*, P_{pub}) \hat{e}(\sum_{i=1}^n P_i^*, V^*) \hat{e}(\sum_{i=1}^n h_i^* P_i^*, W^*) \hat{e}(\sum_{i=1}^n R_i^*, T^*)$. Otherwise, $\mathcal{C}$ aborts.

If $\mathcal{C}$ does not abort, $\mathcal{C}$ compute $abP = (\alpha_{1,0}'^* + h_1^* \alpha_{1,1}'^*)^{-1} (S^* - \sum_{i=2}^n \alpha_{i,0}^* \psi(P_{pub}) - \sum_{i=2}^n \alpha_{i,1}^* \psi(P_{pub}) - \gamma^* \sum_{i=1}^n P_i^* - \pi^* \sum_{i=1}^n h_i^* P_i^* - \beta^* \sum_{i=1}^n R_i^* - (\alpha_{1,0}^* + h_1^* \alpha_{1,1}^*) \psi(P_{pub}))$ or $abP = (\gamma'^* x_1^* + \pi^* h_1^* x_1^*)^{-1} (S^* - \lambda(\sum_{i=1}^n Q_{i,0}^* + \sum_{i=1}^n Q_{i,1}^*) - \sum_{i=2}^n x_i^* \psi(V^*) - \sum_{i=2}^n h_i^* x_i^* \psi(W^*) - \beta^* \sum_{i=1}^n R_i^* - \gamma^* P_1^* - \pi^* h_i^* P_1^*)$, where $Q_{i,j}^* = H_1(ID_i, j)$, $j \in \{0, 1\}$.

Now we compute the probability of C solving the given instance of co-CDH problem. The following four events are considered necessary for the success of for $\mathcal{C}$: $E1$: $\mathcal{C}$ does not abort as a result of any of $\mathcal{A}_I$'s PPK queries/$\mathcal{A}_{II}$'s SK queries. $E2$: $\mathcal{C}$ does not abort as a result of any of $\mathcal{A}_I/\mathcal{A}_{II}$'s SignCrypt queries. $E3$: $\mathcal{A}_I/\mathcal{A}_{II}$ forge a valid and nontrivial aggregated ciphertext. $E4$: Event $E3$ occurs, $\Delta^* = \Delta_J$ and there exists $i \in [1, n]$ such that $ID_i^* = ID_I$ and $h_i^* \neq -\alpha_{i,0}'^*/\alpha_{i,1}'^*$ or $P_i^* = P_M, h_i^* \neq -\gamma_i'^*/\pi_i'^*$ (as mentioned previously, we assume $i = 1$). $\mathcal{C}$ succeeds if all of the above four events happen. Obviously, $\epsilon' \geq (1 - \frac{1}{q_1})^{q_{pk}} (1 - \frac{1}{max(q_{pk}, q_1)}) \frac{1}{q_2} (1 - \frac{1}{q_6'}))^{q_s} \times \frac{1}{max(q_{pk}, q_1)} \frac{1}{q_2} (1 - \frac{1}{q_6'}) \epsilon$

Theorem 3: Our anonymous CASS is sender anonymity in the random oracle model under the co-CDH assumption.

The proof of this theorem is basically the same as Theorem 1, expect that the challenge stage. In the challenge stage, the $\mathcal{C}$ first obtains $\{(ID_{i,j}^*, x_{i,j}^*, P_{i,j}^*)\}_{i=1}^n$ where $j = \{0, 1\}$ from L_K. Then, it sets $R_i^* = d_i bP$ with randomly selected $\{d_i \in_R \mathbb{Z}_q^*\}_{i=1}^n$ and selects a random bit μ. $\mathcal{C}$ obtains hash value $\{K_i\}_{i=1}^n$ from H_5 oracle, $(\Delta^*, T^*, \alpha^*)$ from H_2 oracle, (Δ^*, V^*, β^*) from H_3 oracle, and (Δ^*, W^*, π^*) from H_4 oracle. The component c_i^* and s_i^* are set to be $c_i^* = K_i \oplus (m_i^* \| ID_{\mu,i}^* \| P_{ID_{\mu,i}^*})$ and $s_i^* = \lambda H_1(ID_{\mu,i}^*, 0) + x_i^* V^* + h_i(\lambda H_1(ID_{\mu,i}^*, 1) + x_i^* W^*) + \alpha_i^* R_i^*$ respectively, where $h_i = H_6(P_i^* \| P_j^* \| ID_j^* \| R_i^* \| \Delta_i^* \| m_{\mu,i}^*)$. Finally, the $\mathcal{C}$ gives $(\{R_i^*\}_{i=1}^n, \{c_i^*\}_{i=1}^n, S^* = \sum_{i=1}^n s_i^*)$ to $\mathcal{A}_{II}$.

Theorem 4: Our PPAAS satisfies data confidentiality.

Proof: In our scheme, the traffic-related message sent by a vehicle is signcrypted by the *SignCrypt* algorithm of our CASS. According to Theorem 1, no PPT attacker can learn any information about the content of the message. That means only the intended RSU can obtain the traffic-related message. Therefore, our PPAAS holds data confidentiality.

Theorem 5: Our PPAAS satisfies authentication.

Proof: According to Theorem 2, there is no PPT attacker who can forge a valid ciphertext. Therefore, our PPAAS achieves authentication.

Theorem 6: Our PPAAS satisfies efficient conditional privacy.

Proof: According to Theorem 1 and 3, except the receiver (an RSU), no PPT adversary can distinguish whether two messages are from the same sender and extract the sender's real identity. Therefore, if an RSU is semi-trusted, no malicious vehicle or external attacker can violate the privacy of a vehicle. That is, as long as the designated RSU is not corrupted, the pseudonym of the sender will not be available to an attacker. In this case, the vehicle does not need to change its pseudonym. However, we note that RSUs are deployed along the roadsides. An RSU has the risk of being corrupted by an attacker. In this case, obviously, the attacker may learn the pseudonym of a vehicle. However, a vehicle will only stay in the communication range of the corrupted RSU for a short period of time. When the vehicle moves to the communication range of another RSU that is not corrupted by the attacker, the attacker still cannot learn the pseudonym of the vehicle even the pseudonym of the vehicle has not changed. Therefore, the attacker is unlikely to find the location history of a vehicle based on the vehicle's pseudonym. Considering the facts that the RSUs will only be occasionally corrupted and only the designated RSU can access the pseudonym of a vehicle, vehicles in our scheme do not need to update their pseudonyms frequently. Finally, once a vehicle is found to be malicious, the

TA can retrieve its real identity (See Section V-F). Therefore, our PPAAS achieves efficient conditional privacy.

Theorem 7: Our PPAAS is key escrow freeness.

Proof: In our PPAAS, only a vehicle/RSU knows its full private key. Therefore, our PPAAS achieves key escrow freeness.

VIII. Conclusion

We proposed a PPAAS for safety warning system in fog-cloud based VANTEs based on a new anonymous CASS which fulfills data confidentiality, unforgeability, sender anonymity and key escrow freeness. Compared with the existing solutions, our scheme is more efficient than the existing ones in terms of computation, communication and storage costs, and achieves the security goal of efficient conditional privacy for the first time. Simulation results show that our scheme has low latency and suitable for practical applications.

References

[1] L. Zhang, Q. Wu, J. Domingo-Ferrer, B. Qin, and C. Hu, "Distributed aggregate privacy-preserving authentication in VANETs," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 3, pp. 516–526, Mar. 2017.

[2] S. Lv and Y. Liu, "PLVA: Privacy-preserving and lightweight V2I authentication protocol," *IEEE Trans. Intell. Transp. Syst.*, early access, Feb. 24, 2021, doi: 10.1109/TITS.2021.3059638.

[3] L. Zhang, C. Hu, Q. Wu, J. Domingo-Ferrer, and B. Qin, "Privacy-preserving vehicular communication authentication with hierarchical aggregation and fast response," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2562–2574, Aug. 2016.

[4] Y. Wang, Y. Ding, Q. Wu, Y. Wei, B. Qin, and H. Wang, "Privacy-preserving cloud-based road condition monitoring with source authentication in VANETs," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 7, pp. 1779–1790, Jul. 2019.

[5] L. Zhang, X. Meng, K.-K. R. Choo, Y. Zhang, and F. Dai, "Privacy-preserving cloud establishment and data dissemination scheme for vehicular cloud," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 3, pp. 634–647, Jun. 2020.

[6] H. Cheng, M. Shojafar, M. Alazab, R. Tafazolli, and Y. Liu, "PPVF: Privacy-preserving protocol for vehicle feedback in cloud-assisted VANET," *IEEE Trans. Intell. Transp. Syst.*, early access, Oct. 13, 2021, doi: 10.1109/TITS.2021.3117950.

[7] N. R. Tadapaneni, "Role of fog computing in the Internet of Things," *Int. J. Sci. Res. Eng. Trends*, vol. 180, no. 32, pp. 8887–8975, 2019.

[8] L. Nkenyereye, C. H. Liu, and J. Song, "Towards secure and privacy preserving collision avoidance system in 5G fog based Internet of Vehicles," *Future Gener. Comput. Syst.*, vol. 95, pp. 488–499, Jun. 2019.

[9] W. Wang, L. Wu, W. Qu, Z. Liu, and H. Wang, "Privacy-preserving cloud-fog–based traceable road condition monitoring in VANET," *Int. J. Netw. Manage.*, vol. 31, no. 2, p. e2096, Mar. 2021.

[10] M. Li, L. Zhu, and X. Lin, "Privacy-preserving traffic monitoring with false report filtering via fog-assisted vehicular crowdsensing," *IEEE Trans. Services Comput.*, vol. 14, no. 6, pp. 1902–1913, Nov. 2021.

[11] H. A. Khattak, S. U. Islam, I. U. Din, and M. Guizani, "Integrating fog computing with VANETs: A consumer perspective," *IEEE Commun. Standards Mag.*, vol. 3, no. 1, pp. 19–25, Mar. 2019.

[12] H. Li, G. Zhao, L. Qin, H. Aizeke, X. Zhao, and Y. Yang, "A survey of safety warnings under connected vehicle environments," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 5, pp. 2572–2588, May 2021.

[13] M. Azees, P. Vijayakumar, and L. J. Deboarh, "EAAP: Efficient anonymous authentication with conditional privacy-preserving scheme for vehicular ad hoc networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 9, pp. 2467–2476, Sep. 2017.

[14] M. Armbrust *et al.*, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[15] P. Vijayakumar, M. Azees, S. A. Kozlov, and J. J. P. C. Rodrigues, "An anonymous batch authentication and key exchange protocols for 6G enabled VANETs," *IEEE Trans. Intell. Transp. Syst.*, early access, Aug. 4, 2021, doi: 10.1109/TITS.2021.3099488.

[16] H. Artail and N. Abbani, "A pseudonym management system to achieve anonymity in vehicular ad hoc networks," *IEEE Trans. Dependable Secure Comput.*, vol. 13, no. 1, pp. 106–119, Jan. 2016.

[17] H. H. Nie, Y. P. Li, and Q. H. Wu, "A privacy-preserving V2I authentication scheme without certificates," *J. Inf. Sci. Eng.*, vol. 33, no. 4, pp. 1025–1040, 2017.

[18] A. W. Dent, "Aggregate signcyprion," Int. Assoc. Cryptol. Res. (IACR) Cryptol. ePrint Arch., Tech. Rep., 2021, p. 200. [Online]. Available: https://ia.cr/2012/200

[19] J. Grover, A. Jain, S. Singhal, and A. Yadav, "Real-time VANET applications using fog computing," in *Proc. SSIC*, 2018, pp. 683–691.

[20] H. A. Khattak, S. U. Islam, I. U. Din, and M. Guizani, "Integrating fog computing with VANETs: A consumer perspective," *IEEE Commun. Standards Mag.*, vol. 3, no. 1, pp. 19–25, Mar. 2019.

[21] C. Huang, R. Lu, and K.-K. R. Choo, "Vehicular fog computing: Architecture, use case, and security and forensic challenges," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 105–111, Nov. 2017.

[22] Y. Yao, X. Chang, J. Mišić, and V. Mišić, "Reliable and secure vehicular fog service provision," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 734–743, Feb. 2019.

[23] M. R. Hafner, D. Cunningham, L. Caminiti, and D. Del Vecchio, "Cooperative collision avoidance at intersections: Algorithms and experiments," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1162–1175, Sep. 2013.

[24] J. Molina-Gil, P. Caballero-Gil, and C. Caballero-Gil, "Aggregation and probabilistic verification for data authentication in VANETs," *Inf. Sci.*, vol. 262, pp. 172–189, Mar. 2014.

[25] Q. Huang, N. Li, Z. Zhang, and Y. Yang, "Secure and privacy-preserving warning message dissemination in cloud-assisted Internet of Vehicles," in *Proc. IEEE CNS*, Jun. 2019, pp. 1–8.

[26] S. K. Bhoi and P. M. Khilar, "IJS: An intelligent junction selection based routing protocol for VANET to support ITS services," *Int. Scholarly Res. Notices*, vol. 2014, pp. 1–14, Oct. 2014.

[27] X. Xu, K. Liu, K. Xiao, H. Ren, L. Feng, and C. Chen, "Design and implementation of a fog computing based collision warning system in VANETs," in *Proc. IEEE Symp. Product Compliance Eng.-Asia (ISPCE-CN)*, Dec. 2018, pp. 1–6.

[28] *V2X Applications; Part 1: Road Hazard Signalling (RHS) Application Requirements Specification*, document ETSI TS 101 539-1 V1.1.1, Intelligent Transport Systems, 2013.

[29] *V2X Applications; Part 2: Intersection Collision Risk Warning (ICRW) Application Requirements Specification*, document ETSI TS 101 539-2 V1.1.1, Intelligent Transport Systems, 2018.

[30] *V2X Applications; Part 3: Longitudinal Collision Risk Warning (LCRW) Application Requirements Specification*, document ETSI TS 101 539-1 V1.1.1, Intelligent Transport Systems, 2013.

[31] S. S. D. Selvi, S. S. Vivek, J. Shriram, S. Kalaivani, and C. P. Rangan, "Identity based aggregate signcryption schemes," in *Proc. INDOCRYPT*, 2009, pp. 13–16.

[32] H. Lu and Q. Xie, "An efficient certificateless aggregate signcryption scheme from pairings," in *Proc. Int. Conf. Electron., Commun. Control (ICECC)*, Sep. 2011, pp. 132–135.

[33] Z. Eslami and N. Pakniat, "Certificateless aggregate signcryption: Security model and a concrete construction secure in the random Oracle model," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 26, no. 3, pp. 276–286, Sep. 2014.

[34] S. Basudan, X. Lin, and K. Sankaranarayanan, "A privacy-preserving vehicular crowdsensing-based road surface condition monitoring system using fog computing," *IEEE Internet Things J.*, vol. 4, no. 3, pp. 772–782, Jun. 2017.

[35] C. Zhou, Z. Zhao, W. Zhou, and Y. Mei, "Certificateless key-insulated generalized signcryption scheme without bilinear pairings," *Secur. Commun. Netw.*, vol. 2017, pp. 1–17, Aug. 2017.

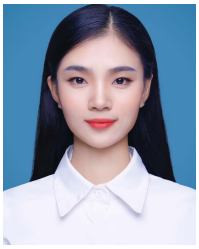
[36] H. Yu and R. Ren, "Certificateless elliptic curve aggregate signcryption scheme," *IEEE Syst. J.*, early access, Aug. 2, 2021, doi: 10.1109/JSYST.2021.3096531.

[37] J. Liu, L. Wang, and Y. Yu, "Improved security of a pairing-free certificateless aggregate signature in healthcare wireless medical sensor networks," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5256–5266, Jun. 2020.

[38] E. R. Magsino and I. W.-H. Ho, "Roadside unit allocation for fog-based information sharing in vehicular networks," in *Proc. 1st ACM Int. Workshop Smart Cities Fog Comput.*, Nov. 2018, pp. 7–12.

[39] B. Cao, Z. Sun, J. Zhang, and Y. Gu, "Resource allocation in 5G IoV architecture based on SDN and fog-cloud computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 6, pp. 3832–3840, Jun. 2021.

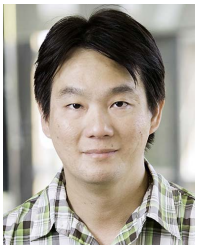
[40] L. Wei, J. Cui, Y. Xu, J. Cheng, and H. Zhong, "Secure and lightweight conditional privacy-preserving authentication for securing traffic emergency messages in VANETs," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 1681–1695, 2021.
[41] J. Liu *et al.*, "Secure intelligent traffic light control using fog computing," *Future Gener. Comput. Syst.*, vol. 78, pp. 817–824, Jan. 2018.
[42] P. Vijayakumar, V. Chang, L. J. Deborah, B. Balusamy, and P. G. Shynu, "Computationally efficient privacy preserving anonymous mutual and batch authentication schemes for vehicular ad hoc networks," *Future Gener. Comput. Syst.*, vol. 78, pp. 943–955, Jan. 2018.
[43] L. Zhang, "OTIBAAGKA: A new security tool for cryptographic mix-zone establishment in vehicular ad hoc networks," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 12, pp. 2998–3010, Dec. 2017.
[44] S. D. Galbraith, K. G. Paterson, and N. P. Smart, "Pairings for cryptographers," *Discrete Appl. Math.*, vol. 156, no. 16, pp. 3113–3121, 2008.
[45] J. Zhang, H. Zhong, J. Cui, Y. Xu, and L. Liu, "SMAKA: Secure many-to-many authentication and key agreement scheme for vehicular networks," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 1810–1824, 2021.



Yafang Yang received the M.S. degree from the College of Mathematics and Statistics, Chongqing University, China, in 2018. She is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Fudan University, China. Her research interests include information security, (certificateless) public key cryptography, applied cryptography, and VANET security.

Lei Zhang (Member, IEEE) received the Ph.D. degree in computer engineering from Universitat Rovira i Virgili, Tarragona, Spain. Since then, he has been with Universitat Rovira i Virgili, as a Post-Doctoral Researcher. He is currently a Full Professor with the Software Engineering Institute, East China Normal University, Shanghai, China. He has been a holder/coholder of more than ten China/Spain-funded (key) projects. His fields of activity are information security, VANET security, cloud security, data privacy, and network security. He has authored over 90 publications. He has served in the program committee of more than 70 international conferences in information security and privacy. He is also an editor of several international journals.

Yunlei Zhao received the Ph.D. degree in computer science from Fudan University, Shanghai, China, in 2004. He joined the Hewlett-Packard European Research Center, Bristol, U.K., as a Post-Doctoral Researcher, in 2004. Since 2005, he has been with Fudan University, where he is currently a Professor with the School of Computer Science. His research interests are the theory and applications of cryptography.



Kim-Kwang Raymond Choo (Senior Member, IEEE) received the Ph.D. degree in information security from the Queensland University of Technology, Australia, in 2006. He currently holds the Cloud Technology Endowed Professorship with The University of Texas at San Antonio (UTSA). He is the Founding Co-Editor-in-Chief of ACM Distributed Ledger Technologies: Research & Practice and the Founding Chair of IEEE TEMS Technical Committee on Blockchain and Distributed Ledger Technologies. He also serves as the Department Editor for IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT, and the Associate Editor for IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, and IEEE TRANSACTIONS ON BIG DATA. He is an ACM Distinguished Speaker and the IEEE Computer Society Distinguished Visitor (2021–2023), a Web of Science's Highly Cited Researcher (Computer Science—2021, Cross-Field—2020), and the recipient of the 2019 IEEE Technical Committee on Scalable Computing Award for Excellence in Scalable Computing (Middle Career Researcher).

Yan Zhang received the B.S. degree (Hons.) from the School of Software Engineering, Jiangxi Normal University, China, and the master's degree from the Software Engineering Institute, East China Normal University, China. Her research interests include VANET security and data privacy.