Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2023.026808

ARTICLE

Heterogeneous Fault-Tolerant Aggregate Signcryption with Equality Test for Vehicular Sensor Networks

Yang Zhao[1], Jingmin An[1], Hao Li[1] and Saru Kumari[2,*]

[1]Network and Data Security Key Laboratory of Sichuan Province, University of Electronic Science and Technology of China, Chengdu, 610054, China

[2]Department of Mathematics, Chaudhary Charan Singh University, Meerut, 250004, India

*Corresponding Author: Saru Kumari. Email: saryusiirohi@gmail.com

Received: 26 September 2022 Accepted: 15 December 2022

ABSTRACT

The vehicular sensor network (VSN) is an important part of intelligent transportation, which is used for real-time detection and operation control of vehicles and real-time transmission of data and information. In the environment of VSN, massive private data generated by vehicles are transmitted in open channels and used by other vehicle users, so it is crucial to maintain high transmission efficiency and high confidentiality of data. To deal with this problem, in this paper, we propose a heterogeneous fault-tolerant aggregate signcryption scheme with an equality test (HFTAS-ET). The scheme combines fault-tolerant and aggregate signcryption, which not only makes up for the deficiency of low security of aggregate signature, but also makes up for the deficiency that aggregate signcryption cannot tolerate invalid signature. The scheme supports one verification pass when all signcryptions are valid, and it supports unbounded aggregation when the total number of signcryptions grows dynamically. In addition, this scheme supports heterogeneous equality test, and realizes the access control of private data in different cryptographic environments, so as to achieve flexibility in the application of our scheme and realize the function of quick search of plaintext or ciphertext. Then, the security of HFTAS-ET is demonstrated by strict theoretical analysis. Finally, we conduct strict and standardized experimental operation and performance evaluation, which shows that the scheme has better performance.

KEYWORDS

Aggregate signcryption; fault-tolerant; heterogeneous; equality test; vehicular sensor network

1 Introduction

In the past few years, the application of Internet of Things (IoT) devices has grown at a great lick, including Industrial Internet of Things (IIoT), intelligent supply chain, electronic medical, smart home and other aspects [1]. Among them, internet of vehicles is one of the most important applications, and the VSN is also one of the key research directions in the academic world. Through wireless communication technology, vehicle equipments effectively use all the dynamic data and information of vehicles in the information network platform. And diverse functional services will be provided by vehicle equipments in the operation control of vehicles.

In the environment of the VSN, massive private data generated by vehicles are transmitted in open channels, such as driving operations inside vehicles, information transmission between vehicles and between vehicles and the Internet. In this case, data confidentiality and transmission efficiency are crucial. At the same time, most IoT rely on cloud computing [2] for massive data processing and services, and strict authentication is required for data use. In this case, data confidentiality and user access are necessary. Therefore, maintaining high transmission efficiency and confidentiality of data is a very important challenge.

In 1976, Diffie [3] researched public key cryptography, and then proposed the concept of digital signature. Digital signature technology combines the identity information of the signer with the signed message, indicating that the signer has signed the message. The verifier can verify that the information is really signed by the signer. Moreover, forging a signature by imitating the signer is difficult. Later, certificate-based signature [4], identity-based signature [5–7] and certificateless signature (CLS) schemes [8–10] emerged successively. However, traditional digital signature has higher computational overhead and lower efficient, so it is not suitable for massive data.

Boneh et al. [11] conducted several research studies to raise verification efficiency and reduce storage capacity. Finally, an aggregate signature scheme was proposed in 2003. This scheme uses the properties of bilinear pairs to generate a short signature, which is more flexible, but it requires different messages to be signed and different participants, which has too many restrictions, higher resource cost and lower availability. Cheon et al. [12] proposed an aggregate signature based identity (IBAS) in 2004. Two certificateless aggregate signature schemes (CLAS) were proposed by Gong et al. [13]. After that, schemes for improvement were put forward in abundance. For example, PFCBAS and CL-DVAAS were proposed respectively by Verma et al. [14] and Deng et al. [15]. These two schemes do not need pair operation, which further improves the verification efficiency. In 2021, Han et al. [16] further improved the existing scheme and proposed an efficient pairing-free CLAS (eCLAS), which reduced the length of signature and the computation cost of verification process. Aggregate signature algorithm is convenient, and it greatly improves the efficiency of verification and effectively reduce the storage capacity. However, aggregate signatures still have two problems: first, the confidentiality of aggregate signatures is low; second, the invalidity of aggregate signatures will lead to the negation of all signatures.

To address the first problem of aggregate signature, Selvi et al. [17] first gave an aggregate signcryption scheme in 2009. Aggregate signcryption aggregates multiple signcrypted ciphertext into a single aggregate signcryption, and the recipient only needs to verify the aggregate signcryption. This means increases the confidentiality of aggregate signature and effectively controls the computation and communication costs [18–21]. In 2011, Lu et al. [22] presented certificateless aggregate signcryption (CLAS), which is based on bilinear mapping and verifiably meets confidentiality and unforgeability. Later, Ren et al. [23] gave a provably secure aggregate signcryption scheme based on identity, which greatly reduced communication overhead, but did not achieve complete aggregation. In 2020, Kim et al. [24] presented a certificateless aggregate signcryption. According to the security requirements and computer resource constraints of IoT, this scheme reduces the computation overhead, communication overhead and storage space, and solves the key aging problem. However, the majority of the existing aggregate signcryption schemes do not support fault tolerance function, and invalid signature will cause that all the aggregated signcrypted ciphertext fail to pass verification.

To address the second issue of aggregate signature, Hartung et al. [25] first presented the concept of fault-tolerant aggregate signatures in 2016, emphasizing fault-tolerant and not mentioning aggregate signature too much, but the scheme was not flexible enough to be applied in practice. In 2019,

Wang et al. [26] proposed an improved fault-tolerant aggregate signature scheme with improved flexibility, but there are still defects, that is, all signatures may still be negated due to a certain or very few invalid signatures. Xiong et al. [27] proposed SECLS, a secure certificateless signature scheme. The scheme supports invalid signature recognition and batch verification. Zhao et al. [28] gave CLFTAS, certificateless fault-tolerant aggregate signature. These two schemes further made up for the defects of fault-tolerant aggregate signature, but when all signatures are valid, the two schemes have higher computation overhead and lower verification efficiency. Xiong et al. [29] proposed an efficient batch verification scheme, while focusing on invalid signature identification. In addition, most aggregate signature schemes do not consider unbounded scheme and do not support the case that the dynamic growth of the total number of signatures.

To ensure the availability of data and searchability, Boneh et al. [30] combined the function of a keyword search with public key encryption, and then presented PKE-KS. This scheme not only ensures data confidentiality, but also ensures data searchability. However, there is a problem: data can be searched if the keyword and data are encrypted using the same public key. Xiong et al. [31], Huang et al. [32] and Chen et al. [33] respectively gave solutions can solve the privacy problems caused by cloud servers (CSs), to achieve access control. Xiong et al. [34] and Mei et al. [35] respectively proposed solutions to the privacy problems of Internet of vehicles and blockchain. In 2010, Yang et al. [36] combined the function of equality test with public key encryption, and then presented PKE-ET. This scheme is not affected by the public key in the encryption process and can carry out equality test discretionarily between ciphertexts. Since then, many scholars have conducted in-depth studies in this field [37–39]. In 2020, Xiong et al. [40] improved this method and applied it to the IIoT environment, which realized data access control in a heterogeneous environment and further improved data security and confidentiality. After that, Xiong et al. [41] ameliorated the scheme further. Xiong et al. [42] revocable scheme and Wu et al. [43] key agreement scheme focus on key security.

For the sake of resolving the above problems, and considering heterogeneity in actual IoT environment, different entities may have different cryptographic environments, so it is necessary to design a fault-tolerant aggregate signcryption scheme that supports a heterogeneous environment.

As shown in Fig. 1, a vehicle in the PKI system (because the vehicle interior is the on-board unit with computing power and communication ability, so the vehicle can be called on-board unit (OBU)) signcrypts message by using the administrator's ID and its own private key to form an individual signcryption, then sends it to the roadside unit (RSU). RSU implements fault-tolerant aggregate of multiple signcryptions and sends aggregate signcryption to CS. At the same time, CS receives trapdoor generated by the administrator in the IBC system. When a user wants to use some data in the IBC system, he encrypts the keywords with his own ID and the corresponding trapdoor, and then sends the encrypted messages to CS. CS determines user's access rights by executing an equality test on the encrypted messages. If the user has right to access these data, CS will return the corresponding data to him.

The detailed contributions of our paper are given below:

(1) The paper constructs a heterogeneous fault-tolerant aggregate signcryption scheme with an equality test (HFTAS-ET). Aggregate signcryption function improves communication data confidentiality and reduces communication overhead. Fault-tolerant function not only tolerates invalid signatures and reduces the verification cost, but also realizes one verification pass when all signcryptions are valid. At the same time, it realizes an unbounded scheme when the number of signcryptions increases dynamically. The scheme supports heterogeneous environment to ensure its flexibility of the scheme, and provides the function of an equality test

to control access rights of data in a heterogeneous environment ensuring the confidentiality and availability of data.

(2) The security of the scheme is verified by strict theoretical analysis. Through detailed functional and performance comparisons, we have concluded that our scheme has better performance and higher efficiency than existing schemes.

(3) This scheme is applicable to the VSN.

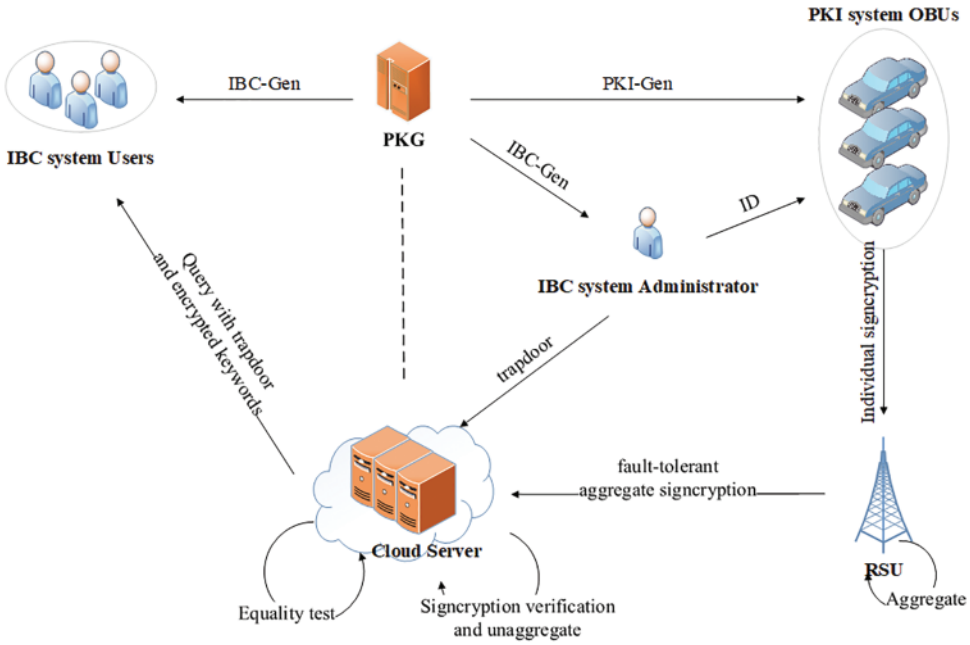

Figure 1: System model

2 Preliminaries

2.1 Bilinear Pairing

Suppose G and G_T are two cyclic groups and their prime orders both are p. P is a generator of G. Define a map $e : G \times G \to G_T$ satisfies the following three conditions:

(1) Bilinearity: $\forall m, n \in G$ and $\forall x, y \in Z_p^*$, there exists $e(xm, yn) = e(m, n)^{xy}$.

(2) Nondegeneracy: $\exists m, n \in G$, such that $e(m, n) \neq 1$.

(3) Computability: $\forall m, n \in G$, there is a viable calculation to compute $e(m, n)$.

2.2 Mathematical Assumption

G and G_T are two cyclic groups and their prime orders are both p. P is a generator of G. There is a bilinear map $e : G \times G \to G_T$. For a random number $x \in Z_p^*$, given (P, xP) to calculate $e(P, P)^{1/x}$ is called **Bilinear Diffie-Hellman Inversion Problem (BDHIP)**.

BDHIA holds if there do not exist probabilistic polynomial-time adversary $\mathscr{A}$ computing BDHIP with probability at least ε. This is called **Bilinear Diffie-Hellman Inversion Assumption (BDHIA)**.

G is a cyclic group and its prime orders is p. P is a generator of G. For a random number $x \in Z_p^*$, given (P, xP) to calculate $(1/x)^P$ is called **Computational Diffie-Hellman Inversion Problem (CDHIP)**.

CDHIP holds if there do not exist probabilistic polynomial-time adversary $\mathscr{A}$ computing CDHIP with probability at least ε. This is called **Computational Diffie-Hellman Inversion Assumption (CDHIA)**.

2.3 Cover-Free Families

In our scheme, d-cover-free families (d-CFFs) is the basis of fault tolerance.

D-cover-Free Family: There exists two sets, one is $\mathscr{X} = \{x_1, \ldots, x_m\}$, where $| \mathscr{X} | = m$, and the other is $\mathscr{D} = \{D_1, \ldots, D_n\}$, where $D_i \subseteq \mathscr{X}$, $1 \leq i \leq n$ and $|\mathscr{D}| = n$. These two sets form a set system $\mathscr{F} = (\mathscr{X}, \mathscr{D})$. A d-cover-free family (d-CFF*(m, n)*) can be represented a set system as follows: $\forall D_{i_0} \in \mathscr{D}$ and other $D_{i_1}, \ldots, D_{i_d} \in \mathscr{D}$, there exists Eq. (1).

$$D_{i_0} \nsubseteq \bigcup_{k=1}^{d} D_{i_k} \tag{1}$$

If the characteristic vectors of subsets in $\mathscr{D}$ are regared as columns of $\mathscr{M}$, then $\mathscr{F}$ can be represented as a binary incidence matrix $\mathscr{M}$ with m rows and n columns. Precisely, if $x_i \in \mathscr{D}$, $\mathscr{M}_{i,k} = 1$, and otherwise $\mathscr{M}_{i,k} = 0$. $\mathscr{M}$ is d-CFF when the corresponding set system is d-CFF.

Nested Family: $(\mathscr{M}^{(\lambda)})_\lambda$ is regarded as a string of incidence matrices of d-CFFs $(\mathscr{F}_\lambda)_\lambda = ((\mathscr{X}_\lambda, \mathscr{D}_\lambda))_\lambda$, and $\mathscr{M}^{(\lambda)}$'s number of rows and columns is $r(\lambda)$ and $c(\lambda)$, respectively.

If $\mathscr{X}_\lambda \subseteq \mathscr{X}_{\lambda+1}$, $r(\lambda) \leq r(\lambda+1)$, $c(\lambda) \leq c(\lambda+1)$, then $\mathscr{M}^{(\lambda+1)} = \begin{pmatrix} \mathscr{M}^{(\lambda)} & \mathscr{Y} \\ \mathscr{Z} & \mathscr{W} \end{pmatrix}$, where $\mathscr{Y}$, $\mathscr{Z}$ and $\mathscr{W}$ are all 0–1 matrices adapted to the size of $\mathscr{M}$, and $\mathscr{Z}$ consists of some rows of $\mathscr{M}^{(\lambda)}$, several rows of all ones and several rows of all zeros, then $(\mathscr{M}^{(\lambda)})_\lambda$ can be regared as a nested family of d-CFFs incidence matrices.

3 System Model

3.1 Formal Definition

Our scheme contains eight algorithms:

(1) **Setup:** It is executed by the private key generator (PKG) according to a number k called security parameter, which generates a collection of system public parameters and master secret key MSK.

(2) **PKI-Gen:** It is executed by PKG according to input a randomly number chosen by a sender s_i in PKI system and further produces the corresponding secret key SK_{s_i} and public key PK_{s_i}.

(3) **IBC-Gen:** It is executed by PKG according to input the ID of a receiver in IBC system and further produces the corresponding secret key SK_r.

(4) **Trapdoor:** Given secret key SK_r as input, the receiver generates the corresponding trapdoor tpd.

(5) **Signcrypt:** It is excuted by OBUs. The sender generates a signcryption σ_i using the sender's secret key SK_{s_i}, a message M_i and the receiver's identity ID for computation.

(6) **Aggregate:** After receiving n senders, n corresponding signcryptions $\{\sigma_i\}_{i=1}^n$, the RSU aggregates all individual signcryptions into a single aggregate signcryption φ by the fault-tolerant aggregate algorithm on the basic of d-CFF.

(7) **Unaggregate:** Given the fault-tolerant aggregate signcryption φ, the secret key of a receiver SK_r, and the public key of n senders $\{SK_{s_i}\}_{i=1}^n$, PKG verifies the signcryption and outputs messages.

(8) **Test:** After receiving signcryption σ_A and trapdoor tpd_A of receiver A, signcryption σ_B and trapdoor tpd_B of the receiver B, CS executed an equality test and produces the corresponding result.

In this scheme, the identity of the administrator is exclusively denoted by ID_{admin}. The scheme is performed as a signcryption scheme and the **Signcrypt** algorithm produces a signcryption of the message M, when $ID = ID_{admin}$. Otherwise, the scheme is performed as a general IBE scheme, the **Signcrypt** algorithm does not run digital signature and only produces encrypted ciphertext of M.

3.2 Security Model

Setup: After obtaining a security parameter k, challenger $\mathscr{C}$ produces the system parameters by executing the **Setup** algorithm. Then, $\mathscr{C}$ performs the **PKI−Gen** algorithm and gets public key and secret key pairs of n senders, $\{(PK_{s_i}^{\#}, SK_{s_i}^{\#})\}_{i=1}^{n}$. Afterward, $\mathscr{C}$ delivers them to adversary $\mathscr{A}_1$.

Phase I: $\mathscr{A}_1$ performs the following queries.

(1) **Key Generation Queries:** After receiving the ID of the required query from $\mathscr{A}_1$, $\mathscr{C}$ executes the **IBC−Gen** algorithm to get the result SK_r, and finally sends it to $\mathscr{A}_1$.

(2) **Aggregate Queries:** After receiving the $\{\sigma_i\}_{i=1}^{n}$ of the required query from $\mathscr{A}_1$, $\mathscr{C}$ executes the **Aggregate** algorithm to get the result φ, and finally sends it to $\mathscr{A}_1$.

(3) **Unaggregate Queries:** After receiving the signcryption φ and receiver's ID of the required query from $\mathscr{A}_1$, $\mathscr{C}$ executes the **Unsigncrypt** algorithm to get the result, and finally sends it to $\mathscr{A}_1$.

Challenge: $\mathscr{A}_1$ sends a receiver's identity $ID^{\#}$ and some message $M_{1,0}^{\#}, M_{1,1}^{\#}, \{M_i\}_{i=2}^{n} \in \{0,1\}^*$ to $\mathscr{C}$. In **Phase I**, $\mathscr{A}_1$ is not allowed to query the secret key of $ID^{\#}$. After that, $\mathscr{C}$ chooses a number $b \in \{0,1\}^*$ at random, and sends $\varphi^{\#}$ to $\mathscr{A}_1$ by executing **Signcrypt** and **Aggregate** algorithm.

Phase II: $\mathscr{A}_1$ is permitted for additional queries in **Phase I**. And the restriction is that the secret key of $ID^{\#}$ and the plaintext of $\varphi^{\#}$ can not be queried during this process.

Guess: $\mathscr{A}_1$ exports its own guess of b'.

Definition 1: If all IND-CCA2 adversaries $\mathscr{A}_1$ with the advantage that $Adv(\mathscr{A}_1) = |2\Pr[b' = b] - 1|$ can be ignored, then our HFTAS-ET scheme is deemed to be IND-CCA2 secure.

Setup: After obtaining a security parameter k, challenger $\mathscr{C}$ produces the system parameters by executing the **Setup** algorithm. Then, $\mathscr{C}$ performs the **PKI−Gen** algorithm and gets public key and secret key pairs of n senders, $\{(PK_{s_i}^{\#}, SK_{s_i}^{\#})\}_{i=1}^{n}$. Afterward, $\mathscr{C}$ delivers them to adversary $\mathscr{A}_2$.

Phase I: $\mathscr{A}_2$ performs the following queries.

(1) **Key Generation Queries:** After receiving the ID of the required query from $\mathscr{A}_2$, $\mathscr{C}$ executes the **IBC−Gen** algorithm to get the result SK_r, and finally sends it to $\mathscr{A}_2$.

(2) **Trapdoor Queries:** After receiving the required query from $\mathscr{A}_2$, $\mathscr{C}$ executes the **Trapdoor** algorithm to get the result tpd, and finally sends it to $\mathscr{A}_2$.

(3) **Aggregate Queries:** After receiving the $\{\sigma_i\}_{i=1}^{n}$ of the required query from $\mathscr{A}_2$, $\mathscr{C}$ executes the **Aggregate** algorithm to get the result φ, and finally sends it to $\mathscr{A}_2$.

(4) **Unaggregate Queries:** After receiving the signcryption φ and receiver's ID of the required query from $\mathscr{A}_2$, $\mathscr{C}$ executes the **Unsigncrypt** algorithm to get the result, and finally sends it to $\mathscr{A}_2$.

Challenge: $\mathscr{A}_2$ sends a receiver's identity $ID^{\#}$ and some message $M_1^{\#}, \{M_i\}_{i=2}^{n} \in \{0,1\}^*$ to $\mathscr{C}$. In **Phase I**, $\mathscr{A}_2$ is not allowed to query the secret key of $ID^{\#}$. After that, $\mathscr{C}$ chooses a number $b \in \{0,1\}^*$ at random, and sends $\varphi^{\#}$ to $\mathscr{A}_2$ by executing **Signcrypt** and **Aggregate** algorithm.

Phase II: $\mathscr{A}_2$ is permitted for additional queries in **Phase I**. And the restriction is that the secret key of $ID^{\#}$ and the plaintext of $\varphi^{\#}$ cannot be queried during this process.

Guess: $\mathscr{A}_2$ exports its own guess of $M_1^{'}$.

Definition 2: If all OW-CCA2 adversaries $\mathscr{A}_2$ with the advantage that $Adv(\mathscr{A}_2) = |Pr[M_1^{'} = M_1^{\#}]|$ can be ignored, then our HFTAS-ET scheme is deemed to be OW-CCA2 secure.

Setup: After obtaining a security parameter k, challenger $\mathscr{C}$ produces the system parameters by executing the **Setup** algorithm. Then, $\mathscr{C}$ performes the **PKI−Gen** algorithm and gets public key of a sender, $PK_s^{\#}$. Afterward, $\mathscr{C}$ delivers it to adversary $\mathscr{A}_3$.

Queries: $\mathscr{A}_3$ performs the following queries:

(1) **Key Generation Queries:** After receiving the ID of the required query from $\mathscr{A}_3$, $\mathscr{C}$ executes the **IBC−Gen** algorithm to get the result SK_r, and finally sends it to $\mathscr{A}_3$.

(2) **Signcryption Queries:** After receiving a plaintext M and a receiver's ID of the required query from $\mathscr{A}_3$, $\mathscr{C}$ executes the **Signcryption** algorithm to get the result σ, and finally sends it to $\mathscr{A}_3$.

Forgery: $\mathscr{A}_3$ exports a receiver's $ID^{\#}$ and a ciphertext of $\sigma^{\#}$ that isn't generated by the oracle of **Signcryption**. $\mathscr{A}_3$ wins if $\sigma^{\#}$ is valid.

Definition 3: If all EUF-CMA adversaries $\mathscr{A}_3$ with the advantage that $Adv(\mathscr{A}_3) = |Pr[\mathscr{A}_3 wins]|$ can be ignored, then our HFTAS-ET scheme is deemed to be EUF-CMA secure.

4 Construction

4.1 The Construction

(1) **Setup:** Given a random number k as security parameter, PKG produces cyclic groups G and G_T, which can be utilized to construct a bilinear map $e : G \times G \to G_T$. P is a generator of G. Calculate $E = e(P, P)$. Choose system master secret key $MSK = (m_1, m_2)$, where $m_1, m_2 \in Z_p^*$. Calculate $P_1 = m_1 P$, $P_2 = m_2 P$. Pick these hash functions: $H_1 : \{0,1\}^* \to Z_p^*$, $H_2 : G_T \times \{0,1\}^* \to Z_p^*$, $H_3 : G_T \to \{0,1\}^*$, $H_4 : G_T \to Z_p^*$, $H_5 : \{0,1\}^* \times Z_p^* \times G_T^3 \to \{0,1\}^*$. The system parameters: $< G, G_T, e, P, P_1, P_2, E, H_1, H_2, H_3, H_4, H_5 >$. Set the special function $F(ID)$, the answer is 1 if $ID = ID_{admin}$, otherwise, the answer is 0.

(2) **PKI-Gen:** PKG input a number $a_{s_i} \in Z_p^*$ randomly chosen by the sender s_i in PKI system and produces the corresponding secret key $SK_{s_i} = \left(1/a_{s_i}\right) P$ and public key $PK_{s_i} = a_{s_i} P$.

(3) **IBC-Gen:** PKG input ID of a receiver in IBC system and produces the corresponding secret key $SK_r = (SK_{r_1}, SK_{r_2})$, where $SK_{r_1} = (1/[H_1(ID) + m_1])\, P$ and $SK_{r_2} = (1/[H_1(ID) + m_2])\, P$.

(4) **Trapdoor:** Input the secret key SK_r of a receiver, and output the corresponding trapdoor $tpd = SK_{r_2}$.

(5) **Signcrypt:** Given the SK_{s_i} of the sender, the plaintext M_i and the ID of a receiver, the sender calculate the corresponding signcryption according to the following steps:

(a) Randomly pick $(u_{1i},\ u_{2i}) \in Z_p^*$.

(b) Set $k_{1i} = E^{u_{1i}}$, $k_{2i} = E^{u_{2i}}$.

(c) Calculate $t_i = H_2(M_i, k_{1i} \cdot k_{2i})$.

Output the ciphertext $\sigma_i = (\alpha_{1i}, \alpha_{2i}, \alpha_{3i}, \alpha_{4i}, \alpha_{5i})$, where $\alpha_{1i} = (M_i||u_{2i}) \oplus H_3(k_{1i})$, $\alpha_{2i} = (u_{2i} \cdot H_1(M_i)) \oplus H_4(k_{2i})$, $\alpha_{3i} = u_{1i}(H_1(ID)P + P_1)$, $\alpha_{4i} = u_{2i}(H_1(ID)P + P_2)$, and $\alpha_{5i} = F(ID)(u_{1i} + t_i)SK_{s_i}$.

(6) **Aggregate:** When receiving n signcryptions $N = \{\sigma_i = (\alpha_{1i}, \alpha_{2i}, \alpha_{3i}, \alpha_{4i}, \alpha_{5i})\}_{i=1}^{n}$ from n senders $\{s_i\}_{i=1}^{n}$ within its coverage, the RSU aggregates all individual signcryptions into a single aggregate signcryption by the following fault tolerant aggregation algorithm:

(a) The α_{5i} part of each ciphertext is extracted and denoted as Q, i.e., $Q = \{\alpha_{51}, \ldots, \alpha_{5n}\}$. Then construct the corresponding binary incidence matrix $\mathscr{M}$ with r rows and n columns, while meeting d-CFF.

(b) In the matrix $\mathscr{M}$, every column represents a signcryption, and every row represents a sub-validation. If $\mathscr{M}_{i,j} = 1$, the i-th sub-validation (i.e., $\varepsilon[i]$) contains the j-th signcryption information α_{5j}. Assuming that $\varepsilon[j]$ is composed by $\{\alpha_{5i}\}_{i=1}^{\omega}$ of $\{\sigma_i\}_{i=1}^{\omega}$, i.e., $\varepsilon[j] = \sum_{i=1}^{\omega} \alpha_{5i}$ for $(1 \le j \le r)$.

(c) Create a new position $\varepsilon[0]$ that satisfies the full aggregation of α_{5i} part in all signcryptions until that signcryption, i.e., $\varepsilon[0] = \sum_{i=1}^{n} \alpha_{5i}$.

(d) The core of aggregate signcryption $\varepsilon = (\varepsilon[0], \varepsilon[1], \ldots, \varepsilon[r])$.

The fault-tolerant aggregate signcryption: $\varphi = (\alpha_{11}, \ldots, \alpha_{1n}, \alpha_{21}, \ldots, \alpha_{2n}, \alpha_{31}, \ldots, \alpha_{3n}, \alpha_{41}, \ldots, \alpha_{4n}, \varepsilon[0], \varepsilon[1], \ldots, \varepsilon[r])$

Unbounded-fault-tolerant aggregate (N_1, N_2)

Let N_1, N_2 are two sets of α_{5i} in two exclusive mergeable signcryptions. Assume that the dimension of N_k is n_k, where $k = 1, 2$ and $n_1 \le n_2$. Let $Q_1 = \{\alpha_{51}, \ldots, \alpha_{5n_1}\}$, $Q_2 = \{\alpha_{51}, \ldots, \alpha_{5n_2}\}$, and corresponding core of aggregate signcryption be $\varepsilon_1 = (\varepsilon_1[0], \varepsilon_1[1], \ldots, \varepsilon_1[r])$, $\varepsilon_2 = (\varepsilon_2[0], \varepsilon_2[1], \ldots, \varepsilon_2[r])$.

Let λ_k satisfies $c(\mathscr{M}^{(\lambda_k - 1)}) < n_k \le c(\mathscr{M}^{(\lambda_k)})$, and $r_k = r(\mathscr{M}^{(\lambda_k)})$ where $k = \{1, 2\}$ and $\lambda_1 \le \lambda_2$. $\mathscr{M}$ is a submatrix of $\mathscr{M}^{(\lambda_2)}$ and made up of the first n_2 columns. Note that $\mathscr{M} = \begin{pmatrix} \mathscr{M}^{(\lambda_1)} & \mathscr{Y} \\ \mathscr{Z} & \mathscr{W} \end{pmatrix}$, for matrices $\mathscr{Y}, \mathscr{Z}, \mathscr{W}$ meeting the "nesting" attribute.

If one or both of the two sets N_k contain only one individual signcryption, ε_k is an individual α_{5k}, then ε_k is expanded into a vector in the manner of Eq. (2), where j is the index of individual signcryption of Q_k.

$$\varepsilon_k[i] = \begin{cases} \alpha_{5k} & i = 0 || (\mathscr{M}[i,j] = 1 \&\& 1 \le i \le r_k) \\ \perp & other \end{cases} \tag{2}$$

Aggregate the corresponding positions of ε_1 and ε_2 based on $\mathscr{M}$, if they are both vectors. Considering special row type of Z, there are three kinds of row index i: *Type* 0(a row of zeros); *Type* 1(a row of ones); *Type* 2(a repeated row r of $\mathscr{M}^{(\lambda_1)}$). Expand ε_1 to make it have the equal dimension as ε_2, which is $\varepsilon_1[i]$ is itself if $1 \le i \le n_1$ and $\varepsilon_1[i] = \perp$ if $n_1 + 1 \le i \le n_2$. After that, execute as the following.

(i) for $i = 0$, aggregate α_{5i} part in all signcryptions to ensure that one verification pass in the case that all signcryptions are valid, as shown in Eq. (3).

$$\varepsilon[0] = \varepsilon_1[0] + \varepsilon_2[0] \tag{3}$$

(ii) for $i = 1, \ldots, r_1$, aggregate the corresponding signcryptions, as shown in Eq. (4).

$$\varepsilon[i] = \varepsilon_1[i] + \varepsilon_2[i] \tag{4}$$

(iii) for $i = r_1 + 1, \ldots, r_2$, as shown in Eq. (5).

$$\varepsilon[i] = \begin{cases} \varepsilon_2[i] & Type\,0 \\ \varepsilon_1[0] + \varepsilon_2[i] & Type\,1 \\ \varepsilon_1[r] + \varepsilon_2[i] & Type\,2 \end{cases} \tag{5}$$

Aggregate with $\left\{\alpha_{1i,N_1}, \alpha_{2i,N_1}, \alpha_{3i,N_1}, \alpha_{4i,N_1}\right\}_{i=1}^{n_1}$ of N_1 and $\left\{\alpha_{1i,N_2}, \alpha_{2i,N_2}, \alpha_{3i,N_2}, \alpha_{4i,N_2}\right\}_{i=1}^{n_2}$ of N_2 to constitute the unbounded-fault-tolerant aggregate signcryption: $\varphi = (\alpha_{11,N_1}, \ldots, \alpha_{1n_1,N_1}, \alpha_{11,N_2}, \ldots, \alpha_{1n_1,N_2}, \alpha_{21,N_1}, \ldots, \alpha_{2n_1,N_1}, \alpha_{21,N_2}, \ldots, \alpha_{2n_1,N_2}, \alpha_{31,N_1}, \ldots, \alpha_{3n_1,N_1}, \alpha_{31,N_2}, \ldots, \alpha_{3n_1,N_2}, \alpha_{41,N_1}, \ldots, \alpha_{4n_1,N_1}, \alpha_{41,N_2}, \ldots, \alpha_{4n_1,N_2}, \varepsilon[0], \varepsilon[1], \ldots, \varepsilon[r_2])$

Output φ.

(7) **Unaggregate:** After receiving the fault-tolerant aggregate signcryption φ, the secretkey of a receiver SK_r, and the public key of n senders $\{SK_{s_i}\}_{i=1}^{n}$, $\varepsilon[j]$ is one of the values ε, where $0 \le j \le r$, and $\varepsilon[j] = \sum_{i=1}^{\omega} \alpha_{51i}$. The algorithm executes as follows:

for $1 \le i \le \omega$,

(a) $k_{1i} = e(\alpha_{3i}, SK_{r_1})$, $k_{2i} = e(\alpha_{4i}, SK_{r_2})$.

(b) $M_i || u_{2i} = \alpha_{1i} \oplus H_3(k_{1i})$.

(c) $t_i = H_2(M_i, k_{1i} \cdot k_{2i})$.

(i) When $F(ID) = 1$, verify if $\alpha_{2i} \oplus (u_{2i} \cdot H_2(M_i)) = H_4(k_{2i})$ and only if $e\left(\Sigma_{i=1}^{\omega} \alpha_{3i}, SK_{r_1}\right) = e\left(\varepsilon[0], \sum_{i=1}^{\omega} PK_{s_i}\right)^{E^{-\sum_{i=1}^{\omega} t_i}}$. If hold, all the signcryptions are valid. Meanwhile, create a new set called "The Valid Set" and add all the signcryptions to it. Then output $\{M_i\}_{i=1}^{n}$. Otherwise, at least one signcryption is invalid.

(ii) When $F(ID) = 0$, verify that $\alpha_{2i} \oplus (u_{2i} \cdot H_2(M_i)) = H_4(k_{2i})$. If it holds, output M_i; if not, output $\perp$.

(8) **Sign:** After receiving a sender's signcryption $\sigma_A = (\alpha_{1A}, \alpha_{2A}, \alpha_{3A}, \alpha_{4A}, \alpha_{5A})$ and a receiver's signcryption $\sigma_B = (\alpha_{1B}, \alpha_{2B}, \alpha_{3B}, \alpha_{4B}, \alpha_{5B})$. The signer calculates the following:

(a) $M'_A = H_5(\alpha_{1A} || \alpha_{2A} || \alpha_{3A} || \alpha_{4A} || \alpha_{5A})$, $M'_B = H_5(\alpha_{1B} || \alpha_{2B} || \alpha_{3B} || \alpha_{4B} || \alpha_{5B})$.

(b) Randomly pick $\left(u'_{1A}, u'_{1B}\right) \in Z_p^*$.

(c) Set $k'_{1A} = E^{u'_{1A}}$, $k'_{1B} = E^{u'_{1B}}$.

(d) Calculate $t_A = H_2(M'_A, k'_{1A})$ and $t_B = H_2(M'_B, k'_{1B})$.

(e) Generate σ_A's signature $\sigma'_A = F(ID)(u'_{1A} + t_A)SK_{s_i}$, σ_B's signature $\sigma'_B = F(ID)(u'_{1B} + t_B)SK_{s_i}$.

(9) **Test:** After receiving a sender's ciphertext $\sigma_A = (\alpha_{1A}, \alpha_{2A}, \alpha_{3A}, \alpha_{4A}, \alpha_{5A})$, the signature σ'_A and the corresponding tpd_A, a receiver's ciphertext $\sigma_B = (\alpha_{1B}, \alpha_{2B}, \alpha_{3B}, \alpha_{4B}, \alpha_{5B})$, the signature σ'_B and the corresponding tpd_B. The algorithm is executed as follows:

(a) Verify if $k'_{1A} = e(\sigma'_A, PK_{s_i})E^{-t_A}, k'_{1B} = e(\sigma'_B, PK_{s_i})E^{-t_B}$. If hold, execute the algorithm according to the procedure below.

(b) $k_{2A} = e(\alpha_{4A}, tpd_A)$, $k_{2B} = e(\alpha_{4B}, tpd_B)$.

(c) $Z_A = \alpha_{2A} \oplus H_4(k_{2A})$, $Z_B = \alpha_{2B} \oplus H_4(k_{2B})$.

(d) Check $k_{2A}^{Z_B} = k_{2B}^{Z_A}$. If it holds, it means that $M_A = M_B$.

4.2 The Identification of Invalid Signcryptions

Given the fault-tolerant aggregate signcryption $\varphi = (\alpha_{11}, \ldots, \alpha_{1n}, \alpha_{21}, \ldots, \alpha_{2n}, \alpha_{31}, \ldots, \alpha_{3n}, \alpha_{41}, \ldots, \alpha_{4n}, \varepsilon[0], \varepsilon[1], \ldots, \varepsilon[r])$, the secretkey of a receiver SK_r, and the public key of n senders $\{SK_{s_i}\}_{i=1}^{n}$, the verification result $e(\Sigma_{i=1}^{\omega}\alpha_{3i}, SK_{r_1}) \neq e(\varepsilon[0], \sum_{i=1}^{\omega} PK_{s_i})^{E^{-\sum_{i=1}^{\omega} t_i}}$.

(1) Verify if $e\left(\Sigma_{i=1}^{\omega}\alpha_{3i}, SK_{r_1}\right) = e\left(\varepsilon[j], \sum_{i=1}^{\omega} PK_{s_i}\right)^{E^{-\sum_{i=1}^{\omega} t_i}}$, for each $1 \leq j \leq r$.

(2) Let inv_e denote the number of the equation does not hold, $1 \leq inv_e \leq r$.

(3) Let inv_s denote the number of invalid signcryption, $1 \leq inv_s \leq n$.

(4) For each ω signcryptions in $\varepsilon[x]$, $1 \leq x \leq inv_e$, verify if $\alpha_{2y} \oplus (u_{2y} \cdot H_2(M_y)) = H_4(k_{2y})$, for $1 \leq y \leq \omega$.

(5) If not hold, this signcryption are not valid. Meanwhile, create a new sete called "The Invalid Set" and add the invalid signcryption to it. Then output M_{iinv_s}. Otherwise, the signcryption is considered valid and appended to "The Valid Set".

5 Security Analysis

Theorem 1: Suppose that BDHIA holds. Our scheme HFTAS-ET is secure against IND-CCA2.

Proof. Suppose there is a challenger $\mathscr{C}$ that can solve BDHIP problem and whose advantage is at least ε. The goal of $\mathscr{C}$ is to compute $e(P, P)^{(1/a)}$, where $a \in Z_p^*$ by knowing an instance (P, aP) of BDHIP. Suppose $\mathscr{A}_1$ can successfully break the HFTAS-ET scheme. A game was placed between challenger $\mathscr{C}$ and adversary $\mathscr{A}_1$. The details of the operation are as given below:

Setup: $\mathscr{C}$ chooses $\theta \in \{1, \ldots, \rho_{H_1}\}$, $L_\theta \in Z_p^*$ and $\lambda_1, \ldots, \lambda_{\theta-1}, \lambda_{\theta+1}, \ldots, \lambda_\rho \in Z_p^*$ at random, where ρ_{H_1} indicates the query times of $\mathscr{H}_1$. Compute $L_i = L_\theta - \lambda_i$, where $i = 1, \ldots, \theta-1, \theta+1, \ldots, \rho$. $\mathscr{C}$ calculate the generator $P \in G_1$ and two parameters $F = aP$, $G = a'P$ by using its input, where $a, a' \in Z_p^*$, and thus it knows $\rho - 1$ pairs $(\lambda_i, U_i = (1/(a + \lambda_i))P)$, $(\lambda_i, V_i = (1/(a' + \lambda_i))P)$ for $i \in \{1, \ldots, \rho\} \setminus \theta$. Choose $P_1 = -F - L_\theta P = (-a - L_\theta)P$ and $P_2 = -G - L_\theta P = (-a' - L_\theta)P$, where s_1 and s_2 are respectively set to $s_1 = -a - L_\theta \in Z_p^*$ and $s_2 = -a' - L_\theta \in Z_p^*$. $(L_i, -U_i) = (L_i, (1/(L_i + s_1))P)$, $(L_i, -V_i) = (L_i, [1/(L_i + \ldots\ldots s_2)]P)$, where $i \in \{1, \ldots, \rho\} \setminus \theta$.

$\mathscr{C}$ sends system parameters, $P_1 = -F - L_\theta P = (-a - L_\theta)\, P$, $P_2 = -G - L_\theta P = (-a' - L_\theta)\, P$, as well as $g = e(P, P)$ to $\mathscr{A}_1$. Afterward, $\mathscr{C}$ returns $\{(PK_{s_i}^{\#}, SK_{s_i}^{\#})\}_{i=1}^{n}$ which is n senders' public/secret-key pair generated by **PKI-Gen** algorithm.

Phase I: $\mathscr{C}$ simulates the original empty $\mathscr{H}_1$, $\mathscr{H}_2$, $\mathscr{H}_3$ and $\mathscr{H}_4$ oracles by preserving L_{H_1}, L_{H_2}, L_{H_3}, and L_{H_4} lists. Assume that each query of $\mathscr{H}_1$ is different, and the identity $ID^{\#}$ is delivered to $\mathscr{H}_1$ at some point. When any other query uses ID, $\mathscr{A}_1$ will query $\mathscr{H}_1(ID)$ in advance. $\mathscr{C}$ responds to $\mathscr{A}_1$ according to the following procedure:

(1) $\mathscr{H}_1$**-Queries:** π indexes these queries, and it is originally set to 1. After receiving a query with ID_π, $\mathscr{C}$ gives L_π and π to $\mathscr{A}_1$. Meanwhile, (ID_π, L_π) is appended to L_{H_1}.

(2) $\mathscr{H}_2$**-Queries:** After receiving a query with (M_i, k_i), $\mathscr{C}$ judges whether (M_i, k_i) exists in L_{H_2}. If so, $\mathscr{C}$ delivers h_{2i} to $\mathscr{A}_1$. Otherwise, $\mathscr{C}$ selects h_{2i} at random in Z_p^* and sends it to $\mathscr{A}_1$. In addition, $\mathscr{C}$ get $h_{3i} = \mathscr{H}(k_{1i})$ and $h_{4i} = \mathscr{H}(k_{2i})$ by simulating the random oracle, where $k_{1i} \cdot k_{2i} = k_i$. Finally, $\mathscr{C}$ calculates $\delta_i = k_{1i} \cdot e(P, P)^{h_{2i}}$ and adds $(M_i, k_i, k_{1i}, k_{2i}, \delta_i, h_{2i})$ into L_{H_2}.

(3) $\mathscr{H}_3$**-Queries:** After receiving a query with k_{1i}, $\mathscr{C}$ judges whether k_{1i} exists in L_{H_3}. If so, $\mathscr{C}$ delivers h_{3i} to $\mathscr{A}_1$. Otherwise, $\mathscr{C}$ selects h_{3i} at random in Z_p^* and sends it to $\mathscr{A}_1$. Meanwhile, (k_{1i}, h_{3i}) is appended to L_{H_3}.

(4) $\mathscr{H}_4$**-Queries:** After receiving a query with k_{2i}, $\mathscr{C}$ judges whether k_{2i} exists in L_{H_4}. If so, $\mathscr{C}$ delivers h_{4i} to $\mathscr{A}_1$. Otherwise, $\mathscr{C}$ selects h_{4i} at random in Z_p^* and sends it to $\mathscr{A}_1$. Meanwhile, (k_{2i}, h_{4i}) is appended to L_{H_4}.

(5) **Key Generation Queries:** After receiving a query with ID_π, $\mathscr{C}$ searches the L_{H_1} list. If $\pi = \theta$, $\mathscr{C}$ aborts. Otherwise, $\mathscr{C}$ knows $\mathscr{H}_1(ID_\pi) = L_\pi$ and delivers $SK_{r_1} = [1/(L_\pi + s_1)]P$, $SK_{r_2} = [1/(L_\pi + s_2)]P$ to $\mathscr{A}_1$.

(6) **Aggregate Queries:** After receiving a query with $\{\sigma_i = (\alpha_{1i}, \alpha_{2i}, \alpha_{3i}, \alpha_{4i}, \alpha_{5i})\}_{i=1}^n$, $\mathscr{C}$ simulates random oracle to obtain $\varepsilon[0] = \sum_{i=1}^n \alpha_{5i}$ and $\varepsilon[j] = \sum_{i=1}^{\omega} \alpha_{5i}$ for $1 \le j \le r$ on the basis of **Aggregate** step, and return $\varphi = (\alpha_{11}, \ldots, \alpha_{1n}, \alpha_{21}, \ldots, \alpha_{2n}, \alpha_{31}, \ldots, \alpha_{3n}, \alpha_{41}, \ldots, \alpha_{4n}, \varepsilon[0], \varepsilon[1], \ldots, \varepsilon[r])$.

(7) **Unaggregate Queries:** When receiving the query with $\varphi = (\alpha_{11}, \ldots, \alpha_{1n}, \alpha_{21}, \ldots, \alpha_{2n}, \alpha_{31}, \ldots, \alpha_{3n}, \alpha_{41}, \ldots, \alpha_{4n}, \varepsilon[0], \varepsilon[1], \ldots, \varepsilon[r])$ and ID_i of a receiver, $\mathscr{C}$ judges whether i equals θ. If not, $\mathscr{C}$ returns $\{M_i\}_{i=1}^n$ based on **Unaggregate**. Otherwise, Eq. (6) holds.

$$\log_{\sum_{i=1}^n SK_{s_i}^*}\left(\varepsilon[0] - \sum_{i=1}^n h_{2i} \cdot SK_{s_i}^{\#}\right) = \log_{(L_iP+P_1)} \sum_{i=1}^n \alpha_{3i} \tag{6}$$

where $h_{2,i} = \mathscr{H}_2(M_i, k_{1i} \cdot k_{2i})$.
Then, $\mathscr{C}$ calculates $\delta = e(\varepsilon[0], L_iP + P_1)$ and searches L_{H_2}. If not found, φ is rejected. Otherwise, $\mathscr{C}$ checks Eq. (7).

$$\frac{e(\sum_{i=1}^n \alpha_{3i}, \sum_{i=1}^n SK_{s_i}^{\#})}{e(L_iP + P_1, \varepsilon[0])} = e\left(L_iP + P_1, \sum_{i=1}^n h_{2i} \cdot SK_{s_i}^{\#}\right) \tag{7}$$

If it holds, return $\{M_i\}_{i=1}^n$; else, for $1 \le j \le r$, $\mathscr{C}$ verifies Eq. (8)

$$\frac{e(\sum_{i=1}^{\omega} \alpha_{3i}, \sum_{i=1}^{\omega} SK_{s_i}^{\#})}{e(L_iP + P_1, \varepsilon[j])} = e\left(L_iP + P_1, \sum_{i=1}^{\omega} h_{2i} \cdot SK_{s_i}^{\#}\right) \tag{8}$$

And return the valid set to $\mathscr{A}_1$.

Challenge: After receiving the receiver's $ID^{\#}$, $M_{1,0}^{\#}$, $M_{1,1}^{\#}$, $\{M_i\}_{i=2}^n \in \{0,1\}^*$, the $\mathscr{C}$ performs algorithm in the following step:

(1) If $ID_i \neq ID^{\#}$, $\mathscr{C}$ will abort.

(2) Otherwise, $\mathscr{C}$ respectively selects b and μ in $\{0,1\}^*$ and Z_p^* at random. $\varphi^{\#} = (\alpha_{11}, \ldots, \alpha_{1n}, \alpha_{21}, \ldots, \alpha_{2n}, \alpha_{31}, \ldots, \alpha_{3n}, \alpha_{41}, \ldots, \alpha_{4n}, \varepsilon[0], \varepsilon[1], \ldots, \varepsilon[r])$ is the ciphertext to be challenged. $\alpha_{1i}, \alpha_{2i} \in \{0,1\}^*$, $\alpha_{3i} = -\mu P$, $\alpha_{4i} \in G_1$, where $1 \le i \le n$. And $\varepsilon[j] \in G_1$, where $1 \le j \le r$. And give $\sigma^{\#}$ to $\mathscr{A}_1$. Let $\kappa = \mu/a$ and $s_1 = -a - L_\theta$, so that for $1 \le i \le n$, we have Eq. (9).

$$\alpha_{3i} = -\mu P = -\kappa a P = (L_\theta + s_1)\kappa P = \kappa L_\theta P + \kappa P_1 \tag{9}$$

Phase II: $\mathscr{A}_1$ is permitted for additional queries in **Phase I**. And the restriction is that the secret key of $ID^{\#}$ and the plaintext of $\varphi^{\#}$ can not be queried during this process.

Guess: $\mathscr{A}_1$ exports its own guess $b' \in \{0,1\}^*$. $\mathscr{C}$ randomly chooses a set $(M_i, k_i, k_{1i}, k_{2i}, \delta_i, h_{2i})$ or (k_{1i}, h_{3i}) from L_{H_2} list or L_{H_3} list and gets $f(y) = \sum_{i=1}^{\rho-1} c_i y^i$ which is a polynomial in $P = f(a)\hat{P}$. Then

outputs $k_{1i} = e(P,P)^{\kappa} = e\left(\hat{P},\hat{P}\right)^{f(a)^2\mu/a}$. If $\delta^{\#} = e\left(\hat{P},\hat{P}\right)^{1/a}$, the BDHIP can be derived via Eq. (10).

$$e(P,P)^{1/a} = \delta^{\#c_0{}^2} e\left(\sum_{t=0}^{\rho-2} c_{t+1}\left(a^t\hat{P}\right), c_0\hat{P}\right) e\left(P, \sum_{t=0}^{\rho-2} c_{t+1}\left(a^t\hat{P}\right)\right) \tag{10}$$

Theorem 2: Suppose that BDHIA holds. Our scheme HFTAS-ET is secure against OW-CCA2.

Proof. Suppose there is a challenger $\mathscr{C}$ that can solve the BDHIP problem and has an advantage is at least ε. The goal of $\mathscr{C}$ is to compute $e(P,P)^{(1/a)}$, where $a \in Z_p^*$ by knowing a instance (P, aP) of BDHIP. Suppose $\mathscr{A}_2$ can successfully break the HFTAS-ET scheme. A game was placed between challenger $\mathscr{C}$ and adversary $\mathscr{A}_2$. The details of the operation are given below:

Setup: $\mathscr{C}$ chooses $\theta \in \{1,\ldots,\rho_{H_1}\}$, $L_\theta \in Z_p^*$ and $\lambda_1,\ldots,\lambda_{\theta-1},\lambda_{\theta+1},\ldots,\lambda_\rho \in Z_p^*$ at random, where ρ_{H_1} indicates the query times of $\mathscr{H}_1$. Compute $L_i = L_\theta - \lambda_i$, where $i = 1,\ldots,\theta-1,\theta+1,\ldots,\rho$. $\mathscr{C}$ calculate the generator $P \in G_1$ and two parameters $F = aP$, $G = a'P$ by using its input, where $a, a' \in Z_p^*$, and thus it konws $\rho - 1$ pairs $(\lambda_i, U_i = (1/(a+\lambda_i))\,P)$, $(\lambda_i, V_i = (1/(a'+\lambda_i))\,P)$ for $i \in \{1,\ldots,\rho\} \setminus \theta$. Choose $P_1 = -F - L_\theta P = (-a - L_\theta)P$ and $P_2 = -G - L_\theta P = (-a' - L_\theta)\,P$, where s_1 and s_2 are respectively set to $s_1 = -a - L_\theta \in Z_p^*$ and $s_2 = -a' - L_\theta \in Z_p^*$. $(L_i, -U_i) = (L_i, (1/(L_i + s_1))\,P)$, $(L_i, -V_i) = (L_i, [1/(L_i + s_2)]\,P)$, where $i \in \{1,\ldots,\rho\} \setminus \theta$.

$\mathscr{C}$ sends system parameters, $P_1 = -F - L_\theta P = (-a - L_\theta)\,P$, $P_2 = -G - L_\theta P = (-a' - L_\theta)\,P$, as well as $g = e(P,P)$ to $\mathscr{A}_2$. Afterward, $\mathscr{C}$ returns $\{(PK_{s_i}^{\#}, SK_{s_i}^{\#})\}_{i=1}^{n}$ which is n senders' public/secret-key pair generated by the **PKI-Gen** algorithm.

Phase I: $\mathscr{C}$ simulates the original empty $\mathscr{H}_1$, $\mathscr{H}_2$, $\mathscr{H}_3$ and $\mathscr{H}_4$ oracles by preserving L_{H_1}, L_{H_2}, L_{H_3}, and L_{H_4} lists. Assume that each query of $\mathscr{H}_1$ is different, and the identity $ID^{\#}$ is delivered to $\mathscr{H}_1$ at some point. When any other query uses ID, $\mathscr{A}_2$ will query $\mathscr{H}_1(ID)$ in advance. $\mathscr{C}$ responds to $\mathscr{A}_2$ according to the following procedure:

(1) $\mathscr{H}_1$**-Queries:** π indexes these queries, and it is originally set to 1. After receiving a query with ID_π, $\mathscr{C}$ gives L_π and π to $\mathscr{A}_2$. Meanwhile, (ID_π, L_π) is appended to L_{H_1}.

(2) $\mathscr{H}_2$**-Queries:** After receiving a query with (M_i, k_i), $\mathscr{C}$ judges whether (M_i, k_i) exists in L_{H_2}. If so, $\mathscr{C}$ delivers h_{2i} to $\mathscr{A}_2$. Otherwise, $\mathscr{C}$ selects h_{2i} at random in Z_p^* and sends it to $\mathscr{A}_2$. In addition, $\mathscr{C}$ get $h_{3i} = \mathscr{H}(k_{1i})$ and $h_{4i} = \mathscr{H}(k_{2i})$ by simulating the random oracle, where $k_{1i} \cdot k_{2i} = k_i$. Finally, $\mathscr{C}$ calculates $\delta_i = k_{1i} \cdot e(P,P)^{h_{2i}}$ and adds $(M_i, k_i, k_{1i}, k_{2i}, \delta_i, h_{2i})$ into L_{H_2}.

(3) $\mathscr{H}_3$**-Queries:** After receiving a query with k_{1i}, $\mathscr{C}$ judges whether k_{1i} exists in L_{H_3}. If so, $\mathscr{C}$ delivers h_{3i} to $\mathscr{A}_2$. Otherwise, $\mathscr{C}$ selects h_{3i} at random in Z_p^* and sends it to $\mathscr{A}_2$. Meanwhile, (k_{1i}, h_{3i}) is appended to L_{H_3}.

(4) $\mathscr{H}_4$**-Queries:** After receiving a query with k_{2i}, $\mathscr{C}$ judges whether k_{2i} exists in L_{H_4}. If so, $\mathscr{C}$ delivers h_{4i} to $\mathscr{A}_2$. Otherwise, $\mathscr{C}$ selects h_{4i} at random in Z_p^* and sends it to $\mathscr{A}_2$. Meanwhile, (k_{2i}, h_{4i}) is appended to L_{H_4}.

(5) **Key Generation Queries:** After receiving a query with ID_π, $\mathscr{C}$ searches the L_{H_1} list. If $\pi = \theta$, $\mathscr{C}$ aborts. Otherwise, $\mathscr{C}$ knows $\mathscr{H}_1(ID_\pi) = L_\pi$ and delivers $SK_{r_1} = [1/(L_\pi + s_1)]P$, $SK_{r_2} = [1/(L_\pi + s_2)]P$ to $\mathscr{A}_2$.

(6) **Trapdoor Queries:** After receiving this query, judge whether π is equal to θ. If so, $\mathscr{C}$ aborts. Otherwise, $\mathscr{C}$ returns $SK_{r_2} = [1/(L_\pi + s_2)]P$ to $\mathscr{A}_2$.

(7) **Aggregate Queries:** After receiving a query with $\{\sigma_i = (\alpha_{1i}, \alpha_{2i}, \alpha_{3i}, \alpha_{4i}, \alpha_{5i})\}_{i=1}^{n}$, $\mathscr{C}$ simulates random oracle to obtain $\varepsilon[0] = \sum_{i=1}^{n} \alpha_{5i}$ and $\varepsilon[j] = \sum_{i=1}^{\omega} \alpha_{5i}$ for $1 \le j \le r$ on the basis of **Aggregate** step, and return $\varphi = (\alpha_{11}, \ldots, \alpha_{1n}, \alpha_{21}, \ldots, \alpha_{2n}, \alpha_{31}, \ldots, \alpha_{3n}, \alpha_{41}, \ldots, \alpha_{4n}, \varepsilon[0], \varepsilon[1], \ldots, \varepsilon[r])$.

(8) **Unaggregate Queries:** When receiving the query with $\varphi = (\alpha_{11}, \ldots, \alpha_{1n}, \alpha_{21}, \ldots, \alpha_{2n}, \alpha_{31}, \ldots, \alpha_{3n}, \alpha_{41}, \ldots, \alpha_{4n}, \varepsilon[0], \varepsilon[1], \ldots, \varepsilon[r])$ and ID_i of a receiver, $\mathscr{C}$ judges whether i equals θ. If not, $\mathscr{C}$ returns $\{M_i\}_{i=1}^{n}$ based on **Unaggregate**. Otherwise, Eq. (11) holds.

$$\log_{\sum_{i=1}^{n} SK_{s_i}^{*}} \left(\varepsilon[0] - \sum_{i=1}^{n} h_{2i} \cdot SK_{s_i}^{\#} \right) = \log_{(L_i P + P_1)} \sum_{i=1}^{n} \alpha_{3i} \tag{11}$$

where $h_{2,i} = \mathscr{H}_2(M_i, k_{1i} \cdot k_{2i})$. Then, $\mathscr{C}$ calculates $\delta = e(\varepsilon[0], L_i P + P_1)$ and searches L_{H_2}. If not found, φ is rejected. Otherwise, $\mathscr{C}$ checks Eq. (12).

$$\frac{e(\sum_{i=1}^{n} \alpha_{3i}, \sum_{i=1}^{n} SK_{s_i}^{\#})}{e(L_i P + P_1, \varepsilon[0])} = e\left(L_i P + P_1, \sum_{i=1}^{n} h_{2i} \cdot SK_{s_i}^{\#} \right) \tag{12}$$

If it holds, return $\{M_i\}_{i=1}^{n}$; else, for $1 \le j \le r$, $\mathscr{C}$ verifies Eq. (13).

$$\frac{e(\sum_{i=1}^{\omega} \alpha_{3i}, \sum_{i=1}^{\omega} SK_{s_i}^{\#})}{e(L_i P + P_1, \varepsilon[j])} = e\left(L_i P + P_1, \sum_{i=1}^{\omega} h_{2i} \cdot SK_{s_i}^{\#} \right) \tag{13}$$

And return the valid set to $\mathscr{A}_2$.

Challenge: After receiving a receiver's $ID^{\#}$, the meaasges $M_1^{\#}, \{M_i\}_{i=2}^{n} \in \{0,1\}^{*}$, $\mathscr{C}$ performs the algorithm in the following step:

(1) If $ID_i \neq ID^{\#}$, $\mathscr{C}$ will abort.

(2) Otherwise, $\mathscr{C}$ respectively selects b and μ in $\{0,1\}^{*}$ and Z_p^{*} at random. $\varphi^{\#} = (\alpha_{11}, \ldots, \alpha_{1n}, \alpha_{21}, \ldots, \alpha_{2n}, \alpha_{31}, \ldots, \alpha_{3n}, \alpha_{41}, \ldots, \alpha_{4n}, \varepsilon[0], \varepsilon[1], \ldots, \varepsilon[r])$ is the ciphertext to be challenged. $\alpha_{1i}, \alpha_{2i} \in \{0,1\}^{*}$, $\alpha_{3i} = -\mu P$, $\alpha_{4i} \in G_1$, where $1 \le i \le n$. And $\varepsilon[j] \in G_1$, where $1 \le j \le r$. And give $\sigma^{\#}$ to $\mathscr{A}_2$. Let $\kappa = \mu / a$ and $s_1 = -a - L_\theta$, so that for $1 \le i \le n$, we have Eq. (14).

$$\alpha_{3i} = -\mu P = -\kappa a P = (L_\theta + s_1)\kappa P = \kappa L_\theta P + \kappa P_1 \tag{14}$$

Phase II: $\mathscr{A}_2$ is permitted for additional queries in **Phase I**. And the restriction is that the secret key of $ID^{\#}$ and the plaintext of $\varphi^{\#}$ can not be queried during this process.

Guess: $\mathscr{A}_2$ exports its own guess $M_1^{'} \in M_1^{\#}$. $\mathscr{C}$ randomly chooses a set $(M_i, k_i, k_{1i}, k_{2i}, \delta_i, h_{2i})$ or (k_{1i}, h_{3i}) from L_{H_2} list or L_{H_3} list and gets $f(y) = \sum_{i=0}^{\rho-1} c_i y^i$ which is a polynomial in $P = f(a)\hat{P}$. Then outputs $k_{1i} = e(P,P)^{\kappa} = e\left(\hat{P}, \hat{P}\right)^{f(a)^2 \mu / a}$. If $\delta^{\#} = e\left(\hat{P}, \hat{P}\right)^{1/a}$, the BDHIP can be derived via Eq. (15).

$$e(P,P)^{1/a} = \delta^{\#{c_0}^2} e\left(\sum_{t=0}^{\rho-2} c_{t+1} \left(a^t \hat{P} \right), c_0 \hat{P} \right) e\left(P, \sum_{t=0}^{\rho-2} c_{t+1} \left(a^t \hat{P} \right) \right) \tag{15}$$

Theorem 3: Suppose that CDHIA holds. Our scheme HFTAS-ET is secure against EUF-CMA.

Proof. Suppose there is a challenger $\mathscr{C}$ that can solve the CDHIP problem and whose advantage is at least ε. The goal of $\mathscr{C}$ is to compute $(1/a)P$, where $a \in Z_p^{*}$ by knowing a instance (P, aP) of CDHIP. Suppose $\mathscr{A}_3$ can successfully break the HFTAS-ET scheme. A game was placed between challenger $\mathscr{C}$ and adversary $\mathscr{A}_3$. The details of the operation are as given below:

Setup: $\mathscr{C}$ obtains system parameters and MSK by performing the **Setup** and then sends the corresponding results to $\mathscr{A}_3$. In addition, $\mathscr{C}$ transmits the sender's public key $PK_s^{\#} = a_i P$ to $\mathscr{A}_3$. $\mathscr{C}$ simulates the original empty $\mathscr{H}_1$, $\mathscr{H}_2$, $\mathscr{H}_3$ and $\mathscr{H}_4$ oracles by preserving every list of L_{H_1}, L_{H_2}, L_{H_3}, and L_{H_4}.

Queries: $\mathscr{C}$ responds to $\mathscr{A}_3$ according to the following procedure:

(1) $\mathscr{H}_1$**-Queries:** After receiving a query with ID_i, $\mathscr{C}$ judges whether ID_i exists in L_{H_1}. If not, $\mathscr{C}$ selects h_{1i} in Z_p^* at random and sends it to $\mathscr{A}_3$. Otherwise, $\mathscr{C}$ delivers h_{1i} to $\mathscr{A}_3$ directly. Meanwhile, (ID_i, h_{1i}) is appended to L_{H_1}.

(2) $\mathscr{H}_2$**-Queries:** After receiving a query with (M_i, k_i), $\mathscr{C}$ judges whether (M_i, k_i) exists in L_{H_2}. If not, $\mathscr{C}$ selects h_{2i} in Z_p^* at random and sends it to $\mathscr{A}_3$. Otherwise, $\mathscr{C}$ delivers h_{2i} to $\mathscr{A}_3$ directly. Meanwhile, $((M_i, k_i), h_{2i})$ is appended to L_{H_2}.

(3) $\mathscr{H}_3$**-Queries:** After receiving a query with k_{1i}, $\mathscr{C}$ judges whether k_{1i} exists in L_{H_3}. If not, $\mathscr{C}$ selects h_{3i} in Z_p^* at random and sends it to $\mathscr{A}_3$. Otherwise, $\mathscr{C}$ delivers h_{3i} to $\mathscr{A}_3$ directly. Meanwhile, (k_{1i}, h_{3i}) is appended to L_{H_3}.

(4) $\mathscr{H}_4$**-Queries:** After receiving a query with k_{2i}, $\mathscr{C}$ judges whether k_{2i} exists in L_{H_4}. If not, $\mathscr{C}$ selects h_{4i} in Z_p^* at random and sends it to $\mathscr{A}_3$. Otherwise, $\mathscr{C}$ delivers h_{4i} to $\mathscr{A}_3$ directly. Meanwhile, (k_{2i}, h_{4i}) is appended to L_{H_4}.

(5) **Key Generation Queries:** After receiving a query with ID_π, $\mathscr{C}$ searches the L_{H_1} list. If $\pi = \theta$, $\mathscr{C}$ aborts. Otherwise, $\mathscr{C}$ knows $\mathscr{H}_1(ID_\pi) = L_\pi$ and delivers $SK_{r_1} = [1/(L_\pi + s_1)]P$, $SK_{r_2} = [1/(L_\pi + s_2)]P$ to $\mathscr{A}_3$.

(6) **Signcryption Queries:** After receiving a query with the ID_i of a receiver and M, $\mathscr{C}$ performs algorithm in the following step:

(a) Randomly pick u_2, μ, $\delta \in Z_p^*$.

(b) Calculate $\alpha_2 = (u_2 \cdot H_1(M)) \oplus H_4(k_{2i})$, $\alpha_5 = \mu SK_{r_{1i}}$, $\alpha_3 = \mu PK_s^{\#} - \delta(H_1(ID_i)P + P_1)$, $\alpha_4 = u_2(H_1(ID_i)P + P_2)$, $k_{1i} = e(\alpha_3, SK_{r_{1i}})$, $k_{2i} = e(\alpha_3, SK_{r_{2i}})$.

(c) Patch the hash value $\mathscr{H}_2(k_{1i} \cdot k_{2i})$ to δ. $\mathscr{C}$ fails if $\mathscr{H}_2$ is defined.

(d) Calculate $\alpha_1 = (M||u_2) \oplus H_3(k_{1i})$
$\mathscr{C}$ returns $\sigma = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5)$ to $\mathscr{A}_3$.

Forgery: According to forking lemma, $\mathscr{A}_3$ can develop a new algorithm $\mathscr{A}_3'$ during the execution. $\mathscr{A}_3$ and $\mathscr{A}_3'$ can export two signatures (M, δ, α_{5i}) and $(M, \delta', \alpha_{5i}')$, where $\delta \neq \delta'$ and k_{1i} are the same for both results. After that, $\mathscr{C}$ can calculate the answer of the CDHIP problem, $(1/a)P = (\delta_i - \delta_i')^{-1}(\alpha_{5i} - \alpha_{5i}')$.

6 Performance Evaluation

In this section, we make a comparison of our scheme and several existing schemes with respect to function comparison, communication and computation overhead.

6.1 Features

In Table 1, we list the functionalities of our scheme compared with the previous similar schemes. From this table, it illustrates that only scheme [40] and our scheme can support heterogeneous signcryption network and have a function of equality test. Among the schemes [14–16,28] that support aggregate signature, only our scheme can support both fault-tolerant aggregation and aggregate signcryption. Compared with the scheme [28] that supports fault-tolerant aggregate signature, our

scheme is an unbounded fault-tolerant aggregate signcryption scheme, which improves efficiency. In addition, our scheme supports one verification pass when all signcryptions are valid, which further improves efficiency.

Table 1: Comparison of functionality

Scheme	PFCBAS [14]	CL-DVAAS [15]	eCLAS [16]	SECLS [27]	CLFTAS [28]	HSC-ET [40]	Ours
Heterogeneous	×	×	×	×	×	√	√
Equality test	×	×	×	×	×	√	√
Signcrypt	×	×	×	×	×	√	√
Aggregation	√	√	√	×	√	×	√
Fault-tolerant	×	×	×	√	√	×	√
Unbounded	×	×	×	×	×	×	√
OVS 1	×	×	×	×	×	×	√

Note: OVS: One verification pass.

6.2 Communication Overhead and Computation Cost

To easily evaluate and analyze the efficiency of our scheme and existing schemes, we use JPBC library to run the experiment on a machine with Windows 10 operating system and Intel Core i7-11700 CPU at 2.50 GHz.

The experimental scheme consists of pairing-based schemes and ECC-based schemes, therefore it is necessary to ensure the same security level. Therefore, two groups are selected, respectively. One is a bilinear pairing $e : G \times G \rightarrow G_T$, where G has order q on a supersingular curve $E : y^2 = x^3 + ax + b \, mod \, p$ and p is a 512-bit prime number. The other is an additive group G' of order q' covering a supersingular elliptic curve $E/F_p : y^2 = x^3 + x \, mod \, p'$, where p, q are two 160-bit prime numbers of 160.

Relevant symbols in this paper are implied in Table 2.

Table 2: The list of notations and descriptions

Symbol	Meaning
$\|G\|$	The size of group G
$\|Z_p\|$	The size of group Z_p
T_{sm-ecc}	The operation of scale multiplication based on elliptic curve
T_{pa-ecc}	The operation of point addition based on elliptic curves
T_p	The operation of pairing
T_{sm}	The operation of scale multiplication on the basis of bilinear pairing
T_{pa}	The operation of point addition on the basis of bilinear pairing
T_h	The operation of hash function
T_e	The operation of exponentiation in G
T_{mi}	A modular inverse in Z_p
T_{mm}	A modular multiplication in Z_p

We contrast our scheme and the previous similar schemes in terms of communication overhead. As shown in Table 3, there is not much difference among these schemes. However, the size of aggregate signcryption of our scheme is significantly less than those of schemes [14–16], while almost the same to scheme [28]. And since our scheme is unbounded aggregation of signcryptions, the small gap is tolerable. Overall, our scheme performs better in terms of communication overhead.

Table 3: Communication overhead comparison

Scheme	The length of secret key	The length of public key	Single signature	Aggregate signature
PFCBAS [14]	$\|Z_p\|$	$\|G\|$	$2\|G\|+4\|Z_p\|$	$2n\|G\|+4\|Z_p\|$
CL-DVAAS [15]	$\|G\|$	$2\|G\|$	$\|G\|+2\|Z_p\|$	$(n+1)\|G\|+n\|Z_p\|$
eCLAS [16]	$\|Z_p\|$	$\|G\|$	$\|G\|+2\|Z_p\|$	$n\|G\|+2\|Z_p\|$
SECLS [27]	$2\|Z_p\|$	$2\|G\|$	$\|G\|+\|Z_p\|$	–
CLFTAS [28]	$\|Z_p\|$	$\|G\|$	$\|G\|+4\|Z_p\|$	$\log_2 n(\|G\|+4\|Z_p\|)$
HSC-ET [40]	$\|G\|$	$\|G\|$	$2\|G\|$	–
Ours	$\|G\|$	$\|G\|$	$2\|G\|$	$2\log_2 n\|G\|$

Table 4 displays a detailed comparison of computation overhead for each phase. Moreover, we perform a detailed comparison experiment as detailed below.

Table 4: Communication overhead comparison

Scheme	Message signature	Signature aggregate	Aggregate verification
PFCBAS [14]	T_{sm-ecc}	$(n-1)T_{pa-ecc}$	$(2n+2)T_{sm-ecc}$
CL-DVAAS [15]	$3T_{sm-ecc}+2T_{pa-ecc}+2T_h$	$(n+3)T_{sm-ecc}$	$(3n+1)T_{sm-ecc}$
eCLAS [16]	$T_{sm-ecc}+T_h$	nT_{pa-ecc}	$(n+1)T_{sm-ecc}+(2n-1)T_{pa-ecc}+nT_h$
SECLS [27]	$T_{sm-ecc}+T_{mi}+2T_{mm}+T_{pa-ecc}$	–	$(3n+1)T_{sm-ecc}+nT_{mm}+2nT_h+2nT_{pa-ecc}$
CLFTAS [28]	$2T_{sm-ecc}+2T_h+2T_{pa-ecc}$	$2nT_{pa-ecc}$	$\log_2 n(4T_{sm-ecc}+5nT_{pa-ecc})$
HSC-ET [40]	$2T_e+2T_h$	–	–
Ours	$2T_e+2T_h$	nT_{pa}	$(2n+2)T_p+3nT_h+T_e$

Fig. 2 shows that time consumption of encryption/signcryption of existing schemes [14–16,27,28,40] and our scheme. Our scheme's signcryption time consumption is slightly higher than that of scheme [14,16], but obviously much lower than that of [15,27,28]. And scheme [14,16] does not need to consider the impact of signcryption stage on fault-tolerant performance in the signature stage, so our scheme has a better signcryption efficiency.

Fig. 3 below shows that time consumption of signatures/signcryptions aggregation in existing schemes [14–16,28] and our scheme. At this stage, our scheme's time consumption is similar to that of [14–16]. In addition, only the scheme [28] and our scheme are fault-tolerant aggregate signature schemes, while the aggregation time of our scheme is far less than the scheme [28].

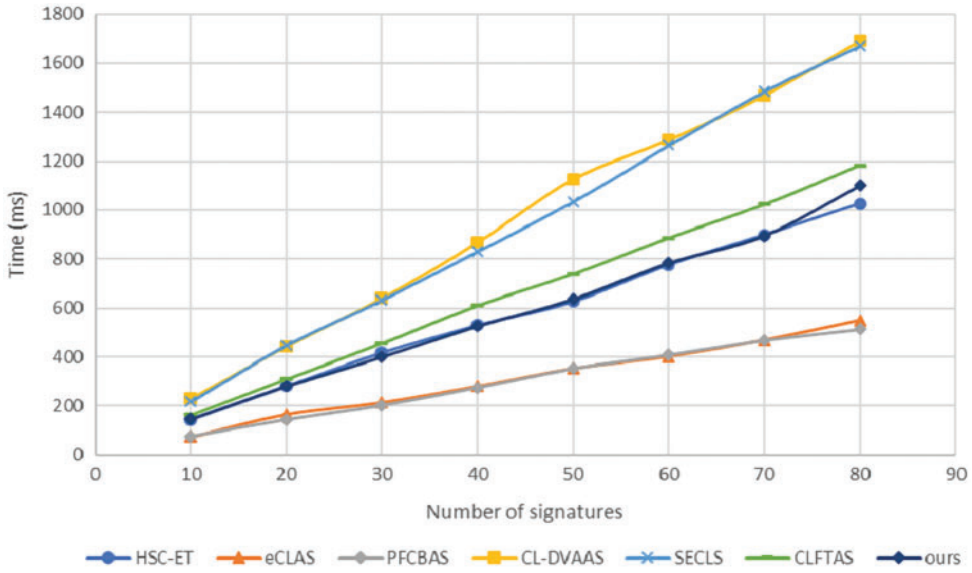

Figure 2: Comparison of encryption (signcryption) cost

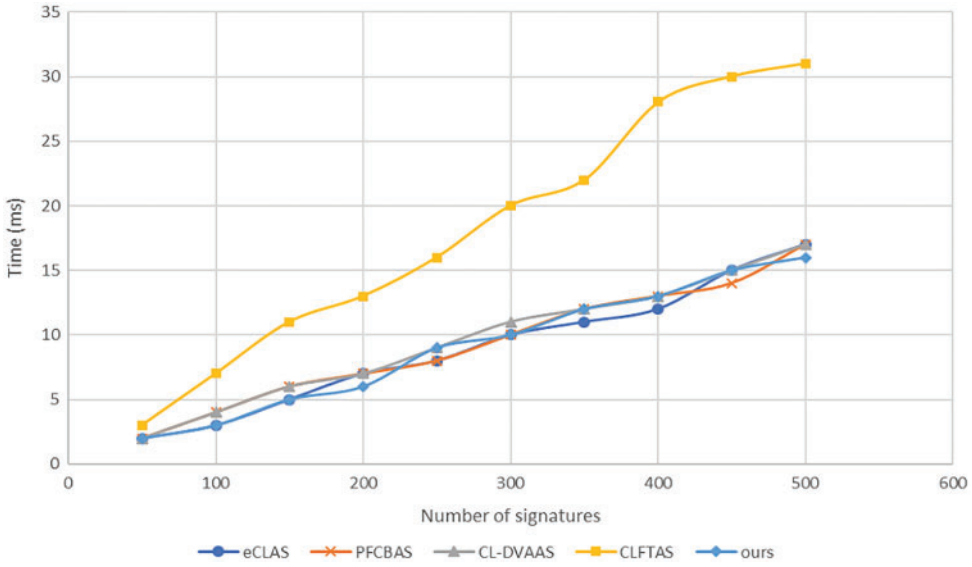

Figure 3: Comparison of signatures (signcryptions) aggregation cost

Fig. 4 below shows that the average time consumption of aggregate signature/signcryption verification in the existing scheme [14–16,28] and our scheme. Our scheme maintains $\log n$ ratio with the number of signatures. Although our scheme has a little more time consumption when there are few signatures, it will be smaller than other existing schemes when the number of signatures increases.

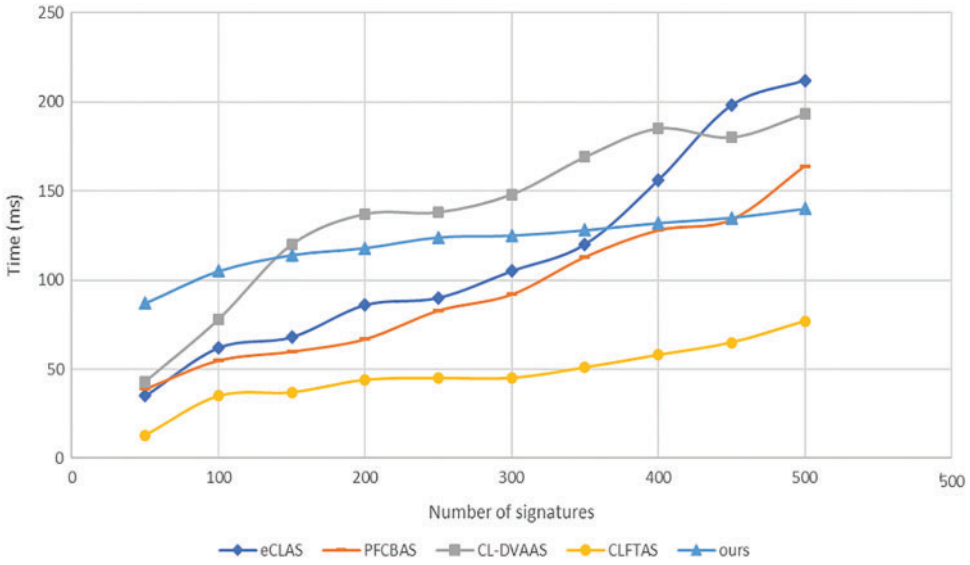

Figure 4: Comparison of aggregate signature (signcryption) verification cost

Finally, we experiment on identifying invalid signatures/signcryptions, as shown in Fig. 5. We assume that signatures/signcryptions n = 100 has an invalid signature/signcryption. In addition, we use binary search method to identify invalid signatures for the scheme [16]. Our scheme supports the identifying of invalid signcryptions information and fault-tolerance, while the scheme [16] only verify the existence of invalid signatures, but cannot tolerate invalid signatures. Therefore, our scheme sacrifices a little verification efficiency, the time consumption of the scheme [16] is less than ours in the best case. But in the worst case, our scheme has less time consumption than the scheme [16].

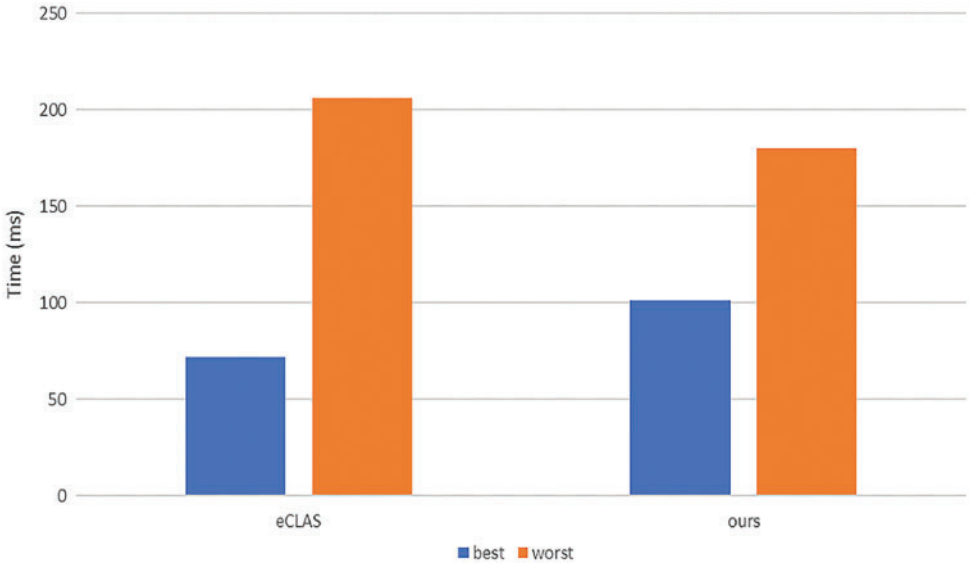

Figure 5: Comparison of identifying invalid signatures (signcryptions)

According to the analysis above, it is clear that our scheme has good performance in message signature/signcryption, aggregate signature/signcryption, aggregate signature/signcryption verification and invalid signature/signcryption identification.

7 Conclusion

In this paper, we give a heterogeneous fault-tolerant aggregate signcryption scheme with equality test, and apply it to the VSN. The scheme adds an unbounded-fault-tolerant function on the basis of aggregate signcryption, which not only strengthens the data confidentiality, but also improves the signcryption verification efficiency. At the same time, the equality test can control data access and ensure the confidentiality of data. In addition, we give a security model of the scheme and prove its security. Finally, experimental operation and performance evaluation show that the scheme has better performance.

Funding Statement: This work was supported in part by the Open Fund of Advanced Cryptography and System Security Key Laboratory of Sichuan Province under Grant SKLACSS-202102, in part by the Intelligent Terminal Key Laboratory of Sichuan Province under Grant SCITLAB-1019.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Asghari, P., Rahmani, A. M., Javadi, H. H. S. (2019). Internet of Things applications: A systematic review. *Computer Networks, 148,* 241–261. https://doi.org/10.1016/j.comnet.2018.12.008
2. Esposito, C., Castiglione, A., Martini, B., Choo, K. K. R. (2016). Cloud manufacturing: Security, privacy, and forensic concerns. *IEEE Cloud Computing, 3(4),* 16–22. https://doi.org/10.1109/MCC.2016.79
3. Diffie, W. (1976). New direction in cryptography. *IEEE Transactions on Information Theory, 22,* 472–492.
4. Thompson, M. R., Essiari, A., Mudumbai, S. (2003). Certificate-based authorization policy in a PKI environment. *ACM Transactions on Information and System Security (TISSEC), 6(4),* 566–588. https://doi.org/10.1145/950191.950196
5. Shamir, A. (1984). Identity-based cryptosystems and signature schemes. *Workshop on the Theory and Application of Cryptographic Techniques*, pp. 47–53. Springer, Berlin, Heidelberg.
6. Hess, F. (2002). Efficient identity based signature schemes based on pairings. *International Workshop on Selected Areas in Cryptography*, pp. 310–324. ST Johns, Canada.
7. Paterson, K. G., Schuldt, J. C. N., Paterson, K. G., Schuldt, J. C. N., Batten, L. M. et al. (2006). Efficient identity-based signatures secure in the standard model. *11th Australasian Conference on Information Security and Privacy*, vol. 4058, pp. 207–222. Melbourne, Australia.
8. Al-Riyami, S. S., Paterson, K. G. (2003). Certificateless public key cryptography. *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 452–473. Taipei, Taiwan.
9. Huang, X., Susilo, W., Mu, Y., Zhang, F. (2005). On the security of certificateless signature schemes from Asiacrypt 2003. *International Conference on Cryptology and Network Security*, pp. 13–25. Berlin, Germany.
10. Harn, L., Ren, J., Lin, C. (2009). Design of DL-based certificateless digital signatures. *Journal of Systems and Software, 82(5),* 789–793. https://doi.org/10.1016/j.jss.2008.11.844
11. Boneh, D., Gentry, C., Lynn, B., Shacham, H. (2003). Aggregate and verifiably encrypted signatures from bilinear maps. *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 416–432. Warsaw, Poland.

12. Cheon, J. H., Kim, Y., Yoon, H. J. (2004). A new ID-based signature with batch verification. *Cryptology ePrint Archive*.
13. Gong, Z., Long, Y., Hong, X., Chen, K. (2007). Two certificateless aggregate signatures from bilinear maps. *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007)*, vol. 3, pp. 188–193. Qungdao, China.
14. Verma, G. K., Singh, B., Kumar, N., Kaiwartya, O., Obaidat, M. S. (2019). Pfcbas: Pairing free and provable certificate-based aggregate signature scheme for the e-healthcare monitoring system. *IEEE Systems Journal, 14(2),* 1704–1715. https://doi.org/10.1109/JSYST.4267003
15. Deng, L., Yang, Y., Gao, R. (2021). Certificateless designated verifier anonymous aggregate signature scheme for healthcare wireless sensor networks. *IEEE Internet of Things Journal, 8(11),* 8897–8909. https://doi.org/10.1109/JIOT.2021.3056097
16. Han, Y., Song, W., Zhou, Z., Wang, H., Yuan, B. (2022). eCLAS: An efficient pairing-free certificateless aggregate signature for secure VANET communication. *IEEE Systems Journal, 16(1),* 1637–1648. https://doi.org/10.1109/JSYST.2021.3116029
17. Selvi, S., Vivek, S. S., Shriram, J., Kalaivani, S., Rangan, C. P. (2009). Identity based aggregate signcryption schemes. *International Conference on Cryptology in India*, pp. 378–397. New Delhi, India.
18. Wang, H., Liu, Z., Liu, Z., Wong, D. S. (2016). Identity-based aggregate signcryption in the standard model from multilinear maps. *Frontiers of Computer Science, 10(4),* 741–754. https://doi.org/10.1007/s11704-015-5138-2
19. Yiliang, H., Fei, C. (2015). The multilinear maps based certificateless aggregate signcryption scheme. *2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, pp. 92–99. Shanghai, China.
20. Eslami, Z., Pakniat, N. (2014). Certificateless aggregate signcryption: Security model and a concrete construction secure in the random oracle model. *Journal of King Saud University-Computer and Information Sciences, 26(3),* 276–286. https://doi.org/10.1016/j.jksuci.2014.03.006
21. Chen, J., Ren, X. (2016). A privacy protection scheme based on certificateless aggregate signcryption and masking random number in smart grid. *International Conference on Mechanical Materials and Manufacturing Engineering*, pp. 10–13. Wuhan, China.
22. Lu, H., Xie, Q. (2011). An efficient certificateless aggregate signcryption scheme from pairings. *2011 International Conference on Electronics, Communications and Control (ICECC)*, pp. 132–135. Ningbo, China.
23. Ren, X. Y., Qi, Z. H., Geng, Y. (2012). Provably secure aggregate signcryption scheme. *ETRI Journal, 34(3),* 421–428. https://doi.org/10.4218/etrij.12.0111.0215
24. Kim, T. H., Kumar, G., Saha, R., Alazab, M., Buchanan, W. J. et al. (2020). CASCF: Certificateless aggregated signcryption framework for internet-of-things infrastructure. *IEEE Access, 8,* 94748–94756. https://doi.org/10.1109/Access.6287639
25. Hartung, G., Kaidel, B., Koch, A., Koch, J., Rupp, A. et al. (2016). Fault-tolerant aggregate signatures. *19th IACR International Conference on the Theory and Practice of Public-Key Cryptography (PKC),* vol. 9614, pp. 331–356. Taipei, Taiwan. https://doi.org/10.1007/978-3-662-49384-7_13
26. Wang, G., Cao, Z., Dong, X., Liu, J. (2019). Improved fault-tolerant aggregate signatures. *The Computer Journal, 62(4),* 481–489. https://doi.org/10.1093/comjnl/bxy108
27. Xiong, H., Wu, Y., Su, C., Yeh, K. -H. (2020). A secure and efficient certificateless batch verification scheme with invalid signature identification for the internet of things. *Journal of Information Security and Applications, 53,* 102507. https://doi.org/10.1016/j.jisa.2020.102507
28. Zhao, Y., Dan, G., Ruan, A., Huang, J., Xiong, H. (2021). A certificateless and privacy-preserving authentication with fault-tolerance for vehicular sensor networks. *2021 IEEE Conference on Dependable and Secure Computing (DSC)*, pp. 1–7. Aizuwakamatsu, Japan.

29. Xiong, H., Jin, C., Alazab, M., Yeh, K. H., Wang, H. et al. (2022). On the design of blockchain-based ecdsa with fault-tolerant batch verification protocol for blockchain-enabled iomt. *IEEE Journal of Biomedical and Health Informatics, 26(5),* 1977–1986. https://doi.org/10.1109/JBHI.2021.3112693
30. Boneh, D., Crescenzo, G. D., Ostrovsky, R., Persiano, G. (2004). Public key encryption with keyword search. *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 506–522. Interlaken, Switzerland.
31. Xiong, H., Yang, M., Yao, T., Chen, J., Kumari, S. (2021). Efficient unbounded fully attribute hiding inner product encryption in cloud-aided wbans. *IEEE Systems Journal, 16(4),* 5424–5432. https://doi.org/10.1109/JSYST.2021.3125455
32. Huang, X., Xiong, H., Chen, J., Yang, M. (2021). Efficient revocable storage attribute-based encryption with arithmetic span programs in cloud-assisted Internet of Things. *IEEE Transactions on Cloud Computing,* 1. https://doi.org/10.1109/TCC.2021.3131686
33. Chen, C. M., Tie, Z., Wang, E. K., Khan, M. K., Kumar, S. et al. (2021). Verifiable dynamic ranked search with forward privacy over encrypted cloud data. *Peer-to-Peer Networking and Applications, 14(5),* 2977–2991. https://doi.org/10.1007/s12083-021-01132-3
34. Xiong, H., Chen, J., Mei, Q., Zhao, Y. (2022). Conditional privacy-preserving authentication protocol with dynamic membership updating for vanets. *IEEE Transactions on Dependable and Secure Computing, 19(3),* 2089–2104. https://doi.org/10.1109/TDSC.2020.3047872
35. Mei, Q., Xiong, H., Chen, Y. C., Chen, C. M. (2022). Blockchain-enabled privacy-preserving authentication mechanism for transportation cps with cloud-edge computing. *IEEE Transactions on Engineering Management,* 1–12. https://doi.org/10.1109/TEM.2022.3159311
36. Yang, G., Tan, C. H., Huang, Q., Wong, D. S. (2010). Probabilistic public key encryption with equality test. *Cryptographers' Track at the RSA Conference*, pp. 119–131. San Francisco, CA.
37. Lee, H. T., Ling, S., Seo, J. H., Wang, H. (2016). Semi-generic construction of public key encryption and identity-based encryption with equality test. *Information Sciences, 373,* 419–440. https://doi.org/10.1016/j.ins.2016.09.013
38. Wu, T., Ma, S., Mu, Y., Zeng, S. (2017). Id-based encryption with equality test against insider attack. *Australasian Conference on Information Security and Privacy*, pp. 168–183. Auckland, New zealand.
39. Qu, H., Yan, Z., Lin, X. J., Zhang, Q., Sun, L. (2018). Certificateless public key encryption with equality test. *Information Sciences, 462,* 76–92. https://doi.org/10.1016/j.ins.2018.06.025
40. Xiong, H., Zhao, Y., Hou, Y., Huang, X., Jin, C. et al. (2020). Heterogeneous signcryption with equality test for IIoT environment. *IEEE Internet of Things Journal, 8(21),* 16142–16152. https://doi.org/10.1109/JIOT.2020.3008955
41. Xiong, H., Hou, Y., Huang, X., Zhao, Y., Chen, C. M. (2022). Heterogeneous signcryption scheme from IBC to PKI with equality test for wbans. *IEEE Systems Journal, 16(2),* 2391–2400. https://doi.org/10.1109/JSYST.2020.3048972
42. Xiong, H., Zhou, Z. D., Wang, L. L., Zhao, Z. T., Huang, X. et al. (2022). An anonymous authentication protocol with delegation and revocation for content delivery networks. *IEEE Systems Journal, 16(3),* 4118–4129. https://doi.org/10.1109/JSYST.2021.3113728
43. Wu, T. Y., Wang, T., Lee, Y. Q., Zheng, W., Kumari, S. et al. (2021). Improved authenticated key agreement scheme for fog-driven iot healthcare system. *Security and Communication Networks, 2021.* https://doi.org/10.1155/2021/6658041