Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

A power resource dispatching framework with a privacy protection function in the Power Internet of Things[*]

Shuanggen LIU[†‡], Shuangzi ZHENG, Wenbo ZHANG[†], Runsheng FU
School of Cyberspace Security, Xi'an University of Posts and Telecommunications, Xi'an 710121, China
[†]E-mail: liusgxupt@163.com; zhangwenbo@xupt.edu.cn

Received Oct. 31, 2021; Revision accepted June 14, 2022; Crosschecked July 4, 2022

Abstract: Smart meters in the Power Internet of Things generate a large amount of power data. However, data privacy in the process of calculation, storage, and transmission is an urgent problem to be solved. Therefore, in this paper we propose a power resource dispatching framework (PRDF) with a privacy protection function, which uses a certificateless aggregate signcryption scheme based on cloud-fog cooperation. Using pseudonyms and aggregating users' power data, PRDF not only protects users' privacy, but also reduces the computing cost and communication overhead under traditional cloud computing. In addition, if the control center finds that a user has submitted abnormal data, it can send a request to the user management center to track the real identity of the user. Our scheme satisfies security requirements based on the random oracle model, including confidentiality and unforgeability. Furthermore, we compare our scheme with other certificateless aggregate signcryption schemes by simulations. Simulation results show that compared with traditional methods, our method performs better in terms of the computation cost.

Key words: Power Internet of Things; Cloud-fog cooperation; Elliptic curve; Random oracle model; Certificateless aggregate signcryption
https://doi.org/10.1631/FITEE.2100518 **CLC number:** TP309

1 Introduction

Power Internet of Things (PIoT) is an industrial Internet of Things. It can connect everything with computers in power systems. For example, it can connect users, power grid enterprises, and power generation enterprises with suppliers to generate shared data and to serve users, power grids, power generation suppliers, governments, and society. Based on the deep perception and advanced communication technology, it improves the level of precise control and intelligent dispatching of power grids. Moreover, PIoT promotes the transformation of traditional power systems to an energy Internet. The specific structure is shown in Fig. 1. Power grid intelligence brings great convenience to our lives. However, with the enrichment of smart grid functions and service improvements, some problems also occur. The concurrent access of a large number of terminal devices in the PIoT leads to significant delay and low security. For example, when smart meters are used in the PIoT, the volume of electricity consumption data which is generated by many electricity meters and usually collected during the same period creates higher data storage and processing capacity requirements. Moreover, there is a privacy protection issue when power data is transmitted in smart grids (Jin, 2021; Li HJ and Gao, 2021).

In view of the problem of data computing and storage, cloud computing can gather many computing resources on cloud platforms to form a virtual

[‡] Corresponding author

[*] Project supported by the National Natural Science Foundation of China (No. 62102311) and the Key Research and Development Program of Shaanxi, China (No. 2021NY-211)

ORCID: Shuanggen LIU, https://orcid.org/0000-0002-8188-2820

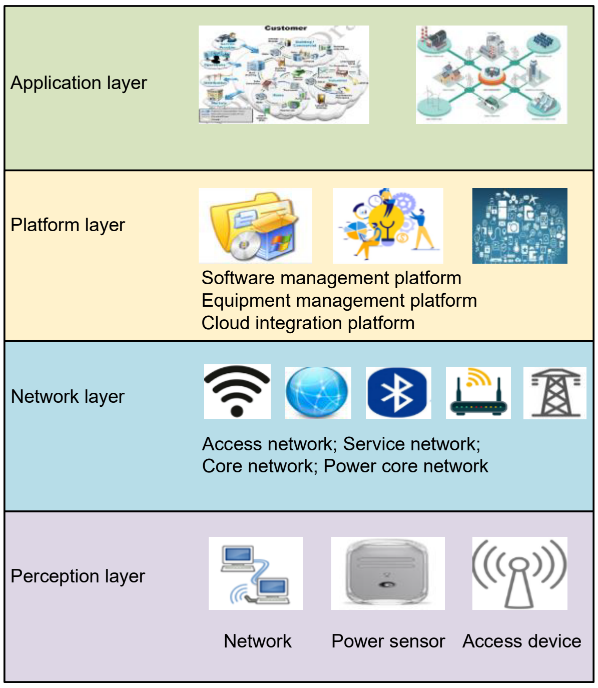
Application layer
Platform layer
Software management platform
Equipment management platform
Cloud integration platform
Network layer
Access network; Service network;
Core network; Power core network
Perception layer
Network
Power sensor
Access device

Fig. 1 Structure of the PIoT

huge computing resource and data center. Users can obtain the required computing and storage resources at a relatively low cost (Cai, 2021; Zhang LY, 2021). However, unacceptable delay caused by long distance transmission of data makes cloud computing unsuitable for delay-sensitive devices. Fog computing is closer to users than traditional cloud computing. By introducing fog layers between the remote cloud layers and terminal devices, fog nodes (FNs) can use batch verification to relieve computing and storage pressure in power grids (Ma B et al., 2019; Ma JJ et al., 2021). It can also reduce the data transmission distance, data transmission delay, and the cost of data sending by terminal devices (Jia and Zhou, 2018; Xu et al., 2018).

1.1 Related works

The common methods for protecting user privacy in the PIoT are anonymity, data aggregation, and adding noise. Most of existing schemes are based on homomorphic encryption (Guo et al., 2020; Shen et al., 2020; Wang XD et al., 2021; Xia et al., 2022). However, homomorphic encryption is not efficient. For resource-constrained devices, more efficient schemes should be considered. Lyu et al. (2018) and Ul Hassan et al. (2019) proposed aggregate schemes using differential privacy. Yu CM et al. (2014) proposed a ring signature scheme in smart grids. Wang L (2019) proposed an aggregate signature scheme, and Wang QY et al. (2020) proposed a batch-verifiable linkable ring signature scheme. They both used digital signatures to achieve integrity and authentication. However, signatures cannot meet confidentiality requirements. User data transmitted in the PIoT should preserve confidentiality and integrity at the same time. Therefore, user data should be encrypted and transmitted in the PIoT. Sui and de Meer (2020) proposed a secure aggregate signcryption scheme based on certificates, which creates a certificate management problem. Chen (2016) proposed a scheme that combines certificateless aggregate signcryption and a masking value, which successfully solves the problem of key escrow. Xie and Li (2020) proposed a certificateless aggregate signcryption scheme with noise. On one hand, the scheme added noise to blur user data. On the other hand, low efficiency operations, such as bilinear pairing and exponential operations, were not used. Consequently, the efficiency of signature verification was improved. However, the scheme could not preserve anonymity or track the real identity of an abnormal user.

Table 1 gives an overview of existing aggregation schemes.

1.2 Motivations

Most of existing certificateless aggregate signcryption schemes are based on bilinear pairing and exponential operations. However, these two operations are much less efficient than scalar multiplication and point addition on elliptic curves. In addition, existing schemes hardly consider the anonymity of every user and the methods for tracking abnormal users.

1.3 Our contributions

To improve the efficiency and protect user privacy, in this paper we propose a power resource dispatching framework (PRDF). The framework uses a certificateless aggregate signcryption scheme with only scalar multiplication and point addition, by which the efficiency is improved. Moreover, users can send data anonymously, which can protect every user's privacy. Our scheme can track the true identity of a user who submits abnormal data, which is of great significance

Table 1 Overview of secure aggregation schemes

Technique	Strength	Weakness
Symmetric homomorphic cryptosystem (Guo et al., 2020)	Lightweight aggregation protocol	Additive operation support; only ciphertext
Paillier cryptosystem (Wang XD et al., 2021)	Support multi-subset data and fault-tolerance	Lack identity authentication and integrity verification
Paillier cryptosystem (Xia et al., 2022)	Supports multi-dimensional data and fault-tolerance	Cannot track abnormal users
Paillier cryptosystem and bilinear pairing (Shen et al., 2020)	Can resist malicious data mining attacks	Control center (CC) can obtain only the total power of the aggregation area
Differential privacy (Ul Hassan et al., 2019)	Introduce a peak factor	Can compute percentage errors and monthly billing
Differential privacy (Lyu et al., 2018)	Fault-tolerance and aggregator obliviousness	Cannot output accurate aggregation results
Bilinear pairing (Sui and de Meer, 2020)	Adopt an aggregation tree to reduce data collector computational cost	Key escrow
Masking value and certificateless technique (Chen, 2016)	Billing function	Cannot output accurate aggregation results
Noise (Xie and Li, 2020)	Use only modular multiplication on elliptic curve	Cannot output accurate aggregation results

for managing users and protecting user privacy. The architecture of cloud-fog cooperation in PIoT is shown in Fig. 2. The main contributions of this paper are as follows:

(1) PRDF manages power data and the real identity of users separately. In this way, it can prevent attackers from directly obtaining the corresponding relationship between the users' identity and their data.

(2) PRDF combines the cloud-fog cooperation mode with certificateless aggregate signcryption technology using pseudonyms. Control center (CC) can analyze power consumption of the whole area and formulate the regional power dispatching strategy without knowing the real identity of the users. Moreover, if a user's data is abnormal, CC will notify the user management center (UMC) to track the abnormal user's real identity.

2 Preparatory knowledge

2.1 Relevant difficult problems

1. Elliptic curve computational Diffie–Hellman problem (ECCDHP)

Let $\mathbb{G}$ be an addition cyclic group of order q, and p be a generator of it. Given aP, $bP\in\mathbb{G}$, for any unknown a, $b\in\mathbb{Z}_q^*$, calculate abP.

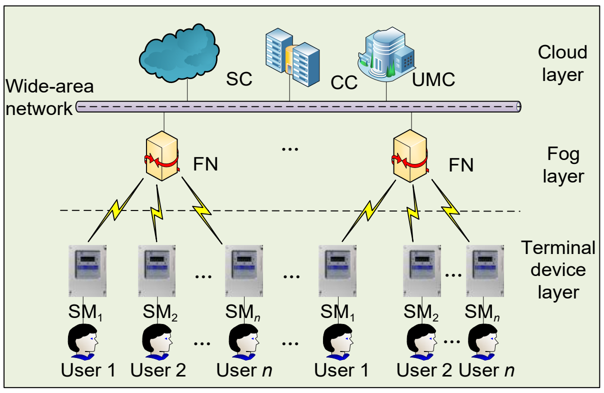

Fig. 2 Architecture of cloud-fog cooperation

2. Elliptic curve discrete logarithm problem (ECDLP)

Let $\mathbb{G}$ be an addition cyclic group of order q on an elliptic curve, and p be a generator of it. Given P, $aP\in\mathbb{G}$, for any unknown $a\in\mathbb{Z}_q^*$, calculate a.

2.2 Formal definition

2.2.1 Frame definition

The certificateless aggregate signcryption scheme in this study consists of the following participants: storage cloud (SC), CC, key generation center (KGC), UMC, FN, and users belonging to the same aggregation area with real identity ID$_i$ (each user has a smart meter SM$_i$). The PRDF model is shown in Fig. 3.

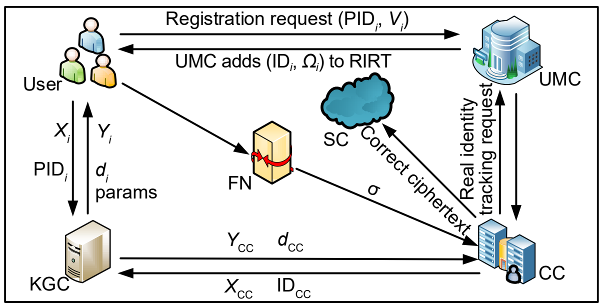
Registration request (PID$_i$, V_i)
UMC adds (ID$_i$, Ω_i) to RIRT
User
UMC
SC
Correct ciphertext
Real identity tracking request
X_i
Y_i
PID$_i$
d_i
params
FN
σ
$Y_{\rm CC}$ $d_{\rm CC}$
KGC
$X_{\rm CC}$ ID$_{\rm CC}$
CC

Fig. 3 Model of the power resource dispatching framework (PRDF)

As shown in Fig. 3, User$_i$ executes the pseudonym generation algorithm to obtain his/her pseudonym PID$_i$ and sends a registration request to UMC. Then, UMC stores (ID$_i$, Ω_i) in the real identity relationship table (RIRT) of SM$_i$. Next, KGC and User$_i$ generate the full keys together. Then, User$_i$ encrypts data m_i, outputs a signature, and sends the signcryption to FN. After receiving the signcryption, FN verifies whether the signature is valid. After passing the verification, FN aggregates the signature and sends the aggregated result to CC, or refuses to accept it. Finally, CC decrypts the ciphertext and verifies the aggregated signature. After verifying the signature successfully, it accepts the decryption result and sends the ciphertext to SC for storage, or refuses to accept it. Moreover, if a user submits abnormal data, CC will notify UMC to track the user's real identity.

2.2.2 Formal definition of certificateless aggregate signcryption scheme

The certificateless aggregate signcryption scheme consists of seven algorithms: system parameter generation, user's key generation, partial key generation, signcryption, aggregation, decryption, and aggregate verification.

(1) System parameter generation: Given a security parameter k, KGC calculates a secret master key s and public parameters.

(2) User's key generation: User$_i$ randomly selects a secret value x_i to calculate public key X_i, and then sends ID$_i$ and X_i to KGC.

(3) Partial key generation: Entering ID$_i$ and X_i, KGC generates the corresponding partial public key and partial private key, and then sends them to User$_i$.

(4) Signcryption: Entering the parameters, message, ID$_i$, private key SK$_i$, public key PK$_i$, and CC's identity ID$_{\rm CC}$ and public key PK$_{\rm CC}$, User$_i$ calculates the signcryption and sends it to FN.

(5) Aggregation: Entering signcryptions σ_i (i=1, 2, $\cdots$, n), FN aggregates signcryptions σ_i into σ and sends σ to CC after the validation of σ_i.

(6) Decryption: Entering parameters, ciphertexts C_i, User$_i$'s public key PK$_i$, ID$_{\rm CC}$, and SK$_{\rm CC}$, CC can obtain messages by decrypting ciphertexts.

(7) Aggregate verification: Entering parameters, aggregated signcryption σ, User$_i$'s public key PK$_i$, ID$_{\rm CC}$, and SK$_{\rm CC}$, CC can verify whether the aggregated signature is valid. If the verification is passed, CC accepts and sends the ciphertexts to SC; otherwise, σ is discarded.

2.3 Pseudonym generation algorithm and real identity tracking algorithm

2.3.1 Pseudonym generation algorithm

When User$_i$ sends a registration request to UMC, Algorithm 1 is executed to generate a pseudonym for User$_i$.

Algorithm 1 Pseudonym generation

```
Input: $H_0$: $\mathbb{G} \rightarrow \mathbb{Z}_q^*$
Output: PID$_i$
1 Select a random number $\Omega_i \in \mathbb{Z}_q^*$
2 Compute $V_i = \Omega_i P$
3 Compute $h_{0i} = H_0(V_i)$
4 Compute PID$_i$ = $h_{0i} \oplus \Omega_i$
5 Send (PID$_i$, $V_i$) to UMC
6 UMC adds (PID$_i$, $V_i$) to its pseudonym relationship
  table (PRT) and adds (ID$_i$, $\Omega_i$) to the real identity rela-
  tionship table (RIRT) of SM$_i$
```

2.3.2 Real identity tracking algorithm

When there are some errors in the data of User$_i$, CC sends a real identity tracking request to UMC. Next, UMC executes Algorithm 2 to track the real identity of the abnormal user.

2.4 Safety model

A certificateless aggregate signcryption scheme must satisfy indistinguishability under the adaptive chosen ciphertext attacks and unforgeability under the adaptive chosen message attack security in the random oracle model. Queries relevant to the sender and receiver having the same identities are not allowed in the random oracle model (Yu HF and Ren,

```
Algorithm 2 Real identity tracking
Input: PID_i, V_i
Output: ID_i
    1 Compute $h_{0i} = H_0(V_i)$
    2 Compute $\Omega_i = \mathrm{PID}_i \oplus h_{0i}$
    3 Search $\mathrm{ID}_i$ from $(\mathrm{ID}_i, \Omega_i)$ in RIRT and sends $\mathrm{ID}_i$ to CC
```

2022). In this study, all entities are semi-honest and may try to infer some useful information of users. The security model of our scheme contains two types of attackers: A_1 and A_2.

A_1 can replace the user's public key, but cannot obtain the system master private key s. It refers to malicious users primarily. The proposed scheme contains A_{11} (attacking the confidentiality of our scheme) and A_{12} (attacking unforgeability of our scheme).

A_2 can obtain the system master private key s, but cannot replace the user's public key. It refers to malicious KGC primarily. The proposed scheme contains A_{21} (attacking the confidentiality of our scheme) and A_{22} (attacking the unforgeability of our scheme).

Definition 1 (Confidentiality) Assume that adversaries A_1 and A_2 cannot win Game 1 or Game 2 with a non-negligible advantage. Then the scheme is secure.

Game 1 (Confidentiality under adversary A_{11}) System parameter generation: Entering a security parameter k, challenger C executes the system parameter generation algorithm to generate system parameters "params" and system master key s, and sends params to A_{11}.

Stage 1: query stage

Adversary A_{11} performs polynomial bounded-time queries as follows:

Public key extraction queries: A_{11} enters the user's identity ID_i for inquiry, and C returns the public key PK_i to A_{11}.

Private key extraction queries: A_{11} enters the user's identity ID_i for inquiry, and C returns the private key SK_i to A_{11}.

Public key replacement queries: A_{11} forges a new public key $\mathrm{PK}_i' = (X_i', Y_i')$ to replace the original public key PK_i to A_{11}.

Signcryption queries: A_{11} enters message m_i, signer's identity ID_i, and receiver's identity ID_B for inquiry, and C returns ciphertext σ_i to A_{11}.

Aggregation queries: A_{11} enters power data m_i, ID_i, and ID_B for inquiry. Next, C returns aggregated signcryption σ to A_{11}.

Decryption queries: A_{11} enters σ and ID_B for inquiry. Next, C returns power data m_i to A_{11}.

Stage 2: challenge stage

After sufficient inquiry, A_{11} selects two plaintexts $m_i\,(i{=}0, 1)$ and two user' identities ID_i and ID_B with a equal length. C determines whether ID_B is a challenging object. If it is not, C rejects it; otherwise, C randomly selects $\xi \in \{0, 1\}$ to generate an aggregate signcryption σ to A_{11}. A_{11} queries in polynomial time again adaptively. It is not allowed to execute private key extraction queries or decryption queries for ID_B.

Stage 3: guess stage

A_{11} guesses a value ξ'. If $\xi' = \xi$, A_{11} wins the game.

Game 2 (Confidentiality under enemy A_{12}) The query phase is similar to that in Game 1, except that public key replacement query and private key extraction query for ID_B cannot be performed.

The challenge phase and guess phase are the same in Game 1. Finally, A_{12} wins the game.

Definition 2 (Unforgeability) Assume that polynomial time adversaries A_1 and A_2 cannot win Game 3 or Game 4 with a non-negligible advantage. Then the scheme is secure.

Game 3 (Unforgeability under enemy A_{21}) The system parameter generation stage and inquiry stage are the same as those in Game 1.

Forgery stage: After the previous two stages, A_{21} outputs a forgery signature σ^*. At least one user ID_i^* does not execute private key extraction queries and ID_B does not execute signcryption queries. Then, A_{21} wins the game.

Game 4 (Unforgeability under enemy A_{22}) The system parameter generation and query phases are the same as those in Game 2. The forgery phase is the same as that in Game 3. Finally, A_{22} wins the game.

3 Concrete scheme

3.1 Description of symbols

Symbols used in this paper are described in Table 2.

3.2 Scheme description

(1) System parameter generation

KGC executes Algorithm 3. KGC selects an elliptic curve $E: y^2 = x^3 + ax + b$. Entering a security

Table 2 Description of symbols

Symbol	Description
s	System master key
P_{pub}	System public key
KGC	Key generation center
H_i	Hash function, i =0, 1, 2, 3
User$_i$	User of the specified aggregation area
ID$_{\text{CC}}$	Identity of CC
SK$_{\text{CC}}$	Private key of CC
PK$_{\text{CC}}$	Public key of CC
ID$_i$	Real identity of User$_i$
x_i	Secret value of User$_i$
y_i	Random number of User$_i$
X_i	Public key of User$_i$
Y_i	Partial public key of User$_i$
d_i	Partial private key of User$_i$
SK$_i$	Full private key of User$_i$, SK$_i$ = (x_i, d_i)
PK$_i$	Full public key of User$_i$, PK$_i$ = (X_i, Y_i)
Δ	Unique state information of each cycle
s_i	User$_i$'s signature of his/her power data
σ_i	Signcryption of User$_i$
m_i	Power data of User$_i$
t	Time stamp
C_i	Ciphertext of User$_i$
PID$_i$	Pseudonym of User$_i$
Ω_i	Secret value corresponding to User$_i$'s identity
PRT	Pseudonym relationship table
RIRT	Real identity relationship table
SC	Storage cloud
CC	Control center
UMC	User management center
FN	Fog node
SM$_i$	Smart meter of User$_i$

parameter k, KGC generates two large prime numbers p and q. q is the order of the cyclic group $\mathbb{G}$ on E and p is a generator of $\mathbb{G}$. KGC selects a random number s as the system master key and generates the system parameters params = $(\mathbb{G}, p, q, P, P_{\text{pub}}, H_1, H_2, H_3)$, as shown in Algorithm 3.

(2) User's key generation

User$_i$ executes Algorithm 4.

(3) Partial key generation

KGC executes Algorithm 5. KGC selects a random number $y_i \in \mathbb{Z}_q^*$ and generates partial public key Y_i for User$_i$. In addition, KGC takes Y_i as a parameter to generate partial private key d_i of User$_i$. Finally, KGC sends (Y_i, d_i) to User$_i$. User$_i$'s public key is PK$_i$ = (X_i, Y_i) and his/her private key is SK$_i$ = (x_i, d_i). In this scheme, CC obtains its key in the same way as User$_i$. Its public key is PK$_{\text{CC}}$ = $(X_{\text{CC}}, Y_{\text{CC}})$ and its private key is SK$_{\text{CC}}$ = $(x_{\text{CC}}, d_{\text{CC}})$.

(4) Signcryption

User$_i$ executes Algorithm 6. User$_i$ selects a random number $r_i \in \mathbb{Z}_q^*$ and calculates R_i, U_i, C_i, and s_i. Finally, User$_i$ sends $\sigma_i = (R_i, s_i, C_i, t)$ to FN.

(5) Aggregation

After receiving signcryptions σ_i (i=1, 2, $\cdots$, n), FN verifies the time stamp of each σ_i. If the time stamp is invalid, the ciphertext is discarded; otherwise, FN executes Algorithm 7.

Algorithm 3 System parameter generation

```
Input: $E: y^2 = x^3 + ax + b$
Output: $s$, params
  1 Generate an additive cyclic group $\mathbb{G}$ from $\mathbb{Z}_q^*$ of prime order $q$ with generator $p$
  2 Select secure hash functions:
     $H_1$: $\{0, 1\}^{l_1} \times \mathbb{G} \times \mathbb{G} \to \mathbb{Z}_q^*$,
     $H_2$: $\{0, 1\}^{l_1} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \{0, 1\}^{l_2} \times \{0, 1\}^{l_3} \to \mathbb{Z}_q^*$,
     $H_3$: $\{0, 1\}^{l_1} \times \{0, 1\}^{l_1} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \to \{0, 1\}^{l_2}$,
     where $l_1$ is the length of the user's real identity or pseudonym, $l_2$ is the length of the message or ciphertext, and $l_3$ is the length of the state information in each cycle
  3 Select a random number $s \in \mathbb{Z}_q^*$ as the system master key
  4 Compute the system master public key $P_{\text{pub}} = sP$
  5 Store $s$ secretly
  6 Publish params = $(\mathbb{G}, p, q, P, P_{\text{pub}}, H_1, H_2, H_3)$
```

Algorithm 4 User's key generation

```
Input: $E: y^2 = x^3 + ax + b$
Output: $x_i$, $X_i$
  1 Select a random $x_i \in \mathbb{Z}_q^*$
  2 Compute $X_i = x_i P$
  3 Send (PID$_i$, $X_i$) to KGC
```

Algorithm 5 Partial key generation

```
Input: PID$_i$
Output: $(Y_i, d_i)$
  1 Select a random $y_i \in \mathbb{Z}_q^*$
  2 Compute partial public key $Y_i = y_i P$
  3 Compute $h_{1i} = H_1(\text{PID}_i, X_i, Y_i)$
  4 Compute partial private key $d_i = y_i + sh_{1i}$
  5 Send $(Y_i, d_i)$ securely to User$_i$
```

(6) Decryption

CC takes the pseudonym, public key, and ciphertext as parameters, and then executes Algorithm 8 to obtain $\mathrm{PID}_i||m_i||t$.

(7) Aggregation verification

CC verifies the signature by executing Algorithm 9. If the result is true, the message $\mathrm{PID}_i||m_i||t$ is valid; otherwise, the message is rejected.

Algorithm 6 Signcryption

```
Input: $\mathrm{PID}_i$
Output: $\sigma_i = (R_i, s_i, C_i, t)$
  1 Select a random $r_i \in \mathbb{Z}_q^*$
  2 Compute $R_i = r_i P$
  3 Compute $U_i = r_i(X_{\mathrm{CC}} + Y_{\mathrm{CC}} + P_{\mathrm{pub}} h_{1\mathrm{CC}})$
  4 Compute $h_{3i} = H_3(\mathrm{PID}_i, \mathrm{ID}_{\mathrm{CC}}, X_i, Y_i, U_i, R_i)$
  5 Compute ciphertext $C_i = (\mathrm{PID}_i||m_i||t) \oplus h_{3i}$
  6 Compute $h_{2i} = H_2(\mathrm{PID}_i, X_{\mathrm{CC}}, Y_{\mathrm{CC}}, R_i, C_i, \Delta)$
  7 Compute $s_i = d_i + x_i h_{2i}$
  8 Send $\sigma_i = (R_i, s_i, C_i, t)$ to FN
```

Algorithm 7 Aggregation

```
Input: $\sigma_i = (R_i, s_i, C_i, t)$
Output: $\sigma$
  1 for $i$ =1 to $n$ do
  2     Compute $S = \sum_{i=1}^{n} s_i$
  3     Return aggregated signcryption $\sigma = (R_i, S, C_i, t)$
  4     Send $\sigma$ to CC
  5 end for
```

Algorithm 8 Decryption

```
Input: $X_i, Y_i, \mathrm{PID}_i, X_{\mathrm{CC}}, Y_{\mathrm{CC}}, R_i, C_i$
Output: $\mathrm{PID}_i||m_i||t$
  1 for $i$=1 to $n$ do
  2     Compute $h_{1i} = H_1(\mathrm{PID}_i, X_i, Y_i)$
  3     Compute partial public key $Y_i = y_i P$
  4     Compute $h_{2i} = H_2(\mathrm{PID}_i, X_{\mathrm{CC}}, Y_{\mathrm{CC}}, R_i, C_i, \Delta)$
  5     Compute $U_i = (x_{\mathrm{CC}} + d_{\mathrm{CC}}) R_i$
  6     Compute $h_{3i} = H_3(\mathrm{PID}_i, \mathrm{ID}_{\mathrm{CC}}, X_i, Y_i, U_i, R_i)$
  7     Recover $\mathrm{PID}_i||m_i||t = C_i \oplus h_{3i}$
  8 end for
```

Algorithm 9 Aggregation verification

```
Input: $S$
Output: True or False
  1 if $SP = \sum_{i=1}^{n} Y_i + P_{\mathrm{pub}} \sum_{i=1}^{n} h_{1i} + \sum_{i=1}^{n} X_i h_{2i}$ is true then
  2     Return True
  3 else
  4     Return False
  5 end if
```

4 Proof of correctness, availability, and security

4.1 Proof of correctness and availability

(1) Correctness

$$\begin{aligned} U_i' &= (x_{\mathrm{CC}} + d_{\mathrm{CC}}) R_i \\ &= r_i(x_{\mathrm{CC}} P + y_{\mathrm{CC}} P + sPh_{1\mathrm{CC}}) \\ &= r_i(X_{\mathrm{CC}} + Y_{\mathrm{CC}} + P_{\mathrm{pub}} h_{1\mathrm{CC}}). \end{aligned} \tag{1}$$

(2) Availability

$$s_i P = Y_i + P_{\mathrm{pub}} h_{1i} + X_i h_{2i}, \tag{2}$$

$$\begin{aligned} SP &= \sum_{i=1}^{n} s_i P \\ &= \sum_{i=1}^{n} (d_i + x_i h_{2i}) P \\ &= \sum_{i=1}^{n} (y_i + sh_{1i} + x_i h_{2i}) P \\ &= \sum_{i=1}^{n} Y_i + P_{\mathrm{pub}} \sum_{i=1}^{n} h_{1i} + \sum_{i=1}^{n} X_i h_{2i}. \end{aligned} \tag{3}$$

4.2 Proof of security

In this study, we will prove the confidentiality of the proposed scheme based on ECCDHP and ECDLP under the random oracle model.

Theorem 1 (Confidentiality under adversary A_{11}) In a random oracle model, the unforgeability of the proposed scheme can be broken if adversary A_{11} can win Game 1 in polynomial time with a non-negligible probability ε_{11} (in the game, A_{11} can do q_s signcryption queries and q_{sk} private key extraction queries at most). Then algorithm Q can solve ECCDHP in polynomial time with at least a non-negligible probability $\left(1-\frac{q_{sk}}{2^k}\right)\left(1-\frac{q_3}{2^k}\right)\frac{\varepsilon_{11}}{en(q_s+q_{sk}+1)}$, where e is the base of the natural logarithm and k is a security parameter.

Proof Q is a solver of the ECCDHP. Given the input (P, aP, bP), its goal is to obtain abP when $a, b \in \mathbb{Z}_q^*$ and are unknown. Q uses adversary A_{11} as a challenger for Game 1. After announcing the start of the game, Q executes the system parameter generation algorithm and sends the public parameters params to A_{11}. Let $P_{\mathrm{pub}} = aP$, and let a act as the system master key. In addition, Q maintains lists L_1, L_2, L_3, L_{SK}, and

L_{PK}, which are used to track the inquiries of A_{11} about H_1 queries, H_2 queries, H_3 queries, private key extraction queries, and public key extraction queries, respectively, for the oracle model. Initially, each list is empty.

Stage 1: query stage

Adversary A_{11} performs polynomial bounded-time queries as follows:

H_1 queries: After receiving the queries of A_{11} for H_1, if the corresponding tuple $(\text{PID}_i, X_i, Y_i, h_{1i})$ exists in L_1, Q returns h_{1i} to A_{11}; otherwise, it makes public key extraction queries on PID_i to obtain the corresponding h_{1i}.

H_2 queries: After receiving the queries of A_{11} for H_2, if the corresponding tuple $(\text{PID}_i, X_i, Y_i, R_i, C_i, \Delta, h_{2i})$ exists in L_2, Q returns h_{2i} to A_{11}; otherwise, it chooses $h_{2i} \in \mathbb{Z}_q^*$ satisfying $h_{2i} \notin L_2$. Next, Q adds $(\text{PID}_i, X_i, Y_i, R_i, C_i, \Delta, h_{2i})$ to L_2 and returns h_{2i} to A_{11}.

H_3 queries: After receiving the queries of A_{11} for H_3, if the corresponding tuple $(\text{PID}_i, \text{ID}_{\text{CC}}, X_i, Y_i, U_i, R_i, h_{3i})$ exists in L_3, Q returns h_{3i} to A_{11}; otherwise, it chooses $h_{3i} \in \{0,1\}$ satisfying $h_{3i} \notin L_3$. Next, Q adds $(\text{PID}_i, \text{ID}_{\text{CC}}, X_i, Y_i, U_i, R_i, h_{3i})$ to L_3 and returns h_{3i} to A_{11}.

Public key extraction queries: When Q receives public key extraction queries, if $(\text{PID}_i, X_i, Y_i, c_i)$ exists in L_{PK}, Q returns the corresponding public key $\text{PK}_i = (X_i, Y_i)$ to A_{11}; otherwise, Q randomly selects a value $c_i \in \{0, 1\}$ with $\Pr[c_i=1]=\delta=\dfrac{1}{q_{\text{s}} + q_{\text{sk}} + 1}$.

If $c_i = 0$, Q randomly selects $x_i, d_i, h_{1i} \in \mathbb{Z}_q^*$ satisfying $X_i, Y_i \notin L_{\text{PK}}$, computes $X_i = x_i P$, $Y_i = d_i P - P_{\text{pub}} h_{1i}$, and adds $(\text{PID}_i, X_i, Y_i, c_i)$ to L_{PK}. Next, Q returns $\text{PK}_i = (X_i, Y_i)$ to A_{11} and adds (PID_i, x_i, d_i) and $(\text{PID}_i, X_i, Y_i, h_{1i})$ to L_{SK} and L_1, respectively.

If $c_i = 1$, Q sets $X_i = r_{\text{know1}} P$ and $Y_i = r_{\text{know2}} P$, where $r_{\text{know1}}, r_{\text{know2}} \in \mathbb{Z}_q^*$ are known random numbers of Q satisfying $X_i, Y_i \notin L_{\text{PK}}$. Next, Q adds $(\text{PID}_i, X_i, Y_i, c_i)$ to L_{PK} and returns $\text{PK}_i = (X_i, Y_i)$ to A_{11}.

Private key extraction queries: Q maintains list L_{SK} with the structure (PID_i, x_i, d_i). When Q receives a query from PID_i, if the corresponding tuple exists in L_{SK}, it returns $\text{SK}_i = (x_i, d_i)$ to A_{11}; otherwise, Q performs public key extraction queries to obtain $(\text{PID}_i, X_i, Y_i, c_i)$.

If $c_i = 0$, it indicates that Q has added (PID_i, x_i, d_i) to L_{SK} at the public key query stage. Next, Q returns $\text{SK}_i = (x_i, d_i)$ to A_{11}.

Otherwise, the simulation terminates.

Public key replacement queries: A_{11} forges a new public key $\text{PK}_i' = (X_i', Y_i')$ to replace the original public key PK_i.

Signcryption queries: Q looks for $(\text{PID}_B, X_B, Y_B, c_B)$ in L_{PK}.

If $c_B = 1$, the query ends and the simulation terminates. Otherwise, Q looks for the private key $\text{SK}_i = (x_i, d_i)$ of PID_i and the public key $\text{PK}_B = (X_B, Y_B)$ of PID_B in L_{SK} and L_{PK}, respectively.

Q executes the signcryption algorithm to generate $\sigma_i = (R_i, s_i, C_i, t)$ for A_{11}.

Aggregation queries: Q generates n signcryptions and computes $S = \sum_{i=1}^{n} s_i$ for n users. Q outputs a valid $\sigma = (R_i, S, C_i, t)$ for A_{11}. Moreover, Q verifies the equation $SP = \sum_{i=1}^{n} Y_i + P_{\text{pub}} \sum_{i=1}^{n} h_{1i} + X_i \sum_{i=1}^{n} h_{2i}$. If the verification fails, the simulation is stopped; otherwise, Q returns σ to A_{11}.

Decryption queries: Q queries the corresponding tuple $(\text{PID}_B, X_B, Y_B, c_B)$ of PID_B in L_{PK} when Q receives decryption queries from A_{11}.

If $c_B = 0$, Q searches the corresponding private key $\text{SK}_B = (x_B, d_B)$ and public key $\text{PK}_i = (X_i, Y_i)$ of PID_i in L_{SK} and L_{PK}, respectively. Next, Q computes $U_i = (x_B + d_B) R_i$ and executes the decryption algorithm to obtain $\text{PID}_i \| m_i \| t$ and $h_{2i} = H_2(\text{PID}_i, X_i, Y_i, R_i, C_i, \Delta)$. Finally, Q returns $\text{PID}_i \| m_i \| t$ to A_{11}.

If $c_B = 1$, Q queries h_{1i}, h_{2i}, and h_{3i} of PID_i in L_1, L_2, and L_3 to compute $\text{PID}_i \| m_i \| t = C_i \oplus h_{3i}$, respectively. Q returns $\text{PID}_i \| m_i \| t$ to A_{11} when $s_i P = Y_i + P_{\text{pub}} h_{1i} + X_i h_{2i}$ is satisfied; otherwise, the simulation terminates.

If c_B does not exist, it implies that the public key has been replaced. Q queries L_1, L_2, and L_3 to compute $\text{PID}_i \| m_i \| t = C_i \oplus h_{3i}$. If $SP = \sum_{i=1}^{n} Y_i + P_{\text{pub}} \sum_{i=1}^{n} h_{1i} + \sum_{i=1}^{n} X_i h_{2i}$ holds, Q returns $\{\text{PID}_i \| m_i \| t\}_{i=1}^{n}$ to A_{11}; otherwise, the simulation terminates.

Stage 2: challenge stage

Adversary A_{11} outputs two identities PID_i and PID_B and two equal-length messages m_0 and m_1, where PID_B is a challenger. Next, Q performs public key generation queries on PID_B to obtain $(\mathrm{PID}_B, X_B, Y_B, c_B)$.

If $c_B = 0$, the simulation terminates.

If $c_B = 1$, Q selects $a, h_{1i}^*, h_{2i}^*, h_{3i}^*, s_i^*, t^* \in \mathbb{Z}_q^*$ randomly to compute $R_i^* = aP$ and makes them satisfy $s_i^* P = Y_i + P_{\mathrm{pub}} h_{1i}^* + X_i h_{2i}^*$. Next, it computes $U_i^* = a(X_B + Y_B + P_{\mathrm{pub}} h_{1i}^*) = (r_{\mathrm{know1}} + r_{\mathrm{know2}} + b h_{1i}^*) R_i^*$, $C_i^* = m_\theta \oplus h_{3i}^*$, $\theta \in \{0, 1\}$, and aggregates $\sigma_i^* = (R_i^*, s_i^*, C_i^*, t^*)$ to $\sigma^* = (R_i^*, s^*, C_i^*, t^*)$.

Finally, Q returns σ^* to A_{11}.

Adversary A_{11} performs the above queries for probabilistic polynomial times and outputs guesses on θ', $\theta' \in \{0, 1\}$. If $\theta' = \theta$, Q outputs $abP = \frac{1}{h_{1B}}[U_i^* - R_i^*(r_{\mathrm{known1}} + r_{\mathrm{known2}})]$ as an effective solution to ECCDHP; otherwise, the difficult problem is not solved.

Probabilistic analysis: Q solves ECCDHP successfully, which means that it does not stop the simulation all the time and adversary A_{11} breaks through the confidentiality of the proposed scheme with a non-negligible probability ε_{11}. Q will succeed only if the following events do not occur:

ε_1: At least one user PID_i does not ask for private key generation with a probability of $\Pr[\varepsilon_1] = \frac{1}{n}\left(1 - \frac{q_{\mathrm{sk}}}{2^k}\right)$.

ε_2: Q does not query H_3 with a probability of $\Pr[\varepsilon_2] = 1 - \frac{q_3}{2^k}$.

ε_3: Q does not terminate the query phase with a probability of $\Pr[\varepsilon_3] = (1-\delta)^{q_{\mathrm{s}} + q_{\mathrm{sk}} + 1}$. When $q_{\mathrm{s}} + q_{\mathrm{sk}}$ is large enough, $(1-\delta)^{q_{\mathrm{s}} + q_{\mathrm{sk}} + 1}$ tends to e^{-1}.

ε_4: Q does not terminate the challenge phase with the probability of $\Pr[\varepsilon_4] = \delta$.

Therefore, the confidentiality of this scheme can be broken only if adversary A_{11} solves ECCDHP with a non-negligible probability $\left(1 - \frac{q_{\mathrm{sk}}}{2^k}\right) \cdot \left(1 - \frac{q_3}{2^k}\right) \frac{\varepsilon_{11}}{\mathrm{e}n(q_{\mathrm{s}} + q_{\mathrm{sk}} + 1)}$.

Theorem 2 (Confidentiality under adversary A_{12}) In a random oracle model, the unforgeability of the proposed scheme can be broken if adversary A_{12} can win Game 2 in polynomial time with a non-negligible probability ε_{12} (in the game, A_{12} can execute q_{s} signcryption queries and q_{sk} private key extraction queries at most). Then Q can solve ECCDHP in polynomial time with at least a non-negligible probability $\left(1 - \frac{q_{\mathrm{sk}}}{2^k}\right)\left(1 - \frac{q_3}{2^k}\right) \frac{\varepsilon_{12}}{\mathrm{e}n(q_{\mathrm{s}} + q_{\mathrm{sk}} + 1)}$.

Proof Q is a solver of ECCDHP. Given the input (P, aP, bP), its goal is to obtain abP when $a, b \in \mathbb{Z}_q^*$ and are unknown. Q uses adversary A_{12} as a challenger to the game. After announcing the start of the game, Q executes the system parameter generation algorithm. Next, Q sends the public parameters params and the master key s to A_{12}. In addition, Q maintains L_1, L_2, L_3, L_{SK}, and L_{PK}, which are used to track the inquiries of A_{12} about H_1 queries, H_2 queries, H_3 queries, private key extraction queries, and public key extraction queries for the oracle model, respectively. Initially, each list is empty.

Stage 1: query stage

Adversary A_{12} performs polynomial bounded-time queries for H_1 queries, H_2 queries, H_3 queries, private key extraction queries, public key replacement queries, and signcryption queries in Theorem 1.

Public key extraction queries: when Q receives a public key extraction query, if $(\mathrm{PID}_i, X_i, Y_i, c_i)$ exists in L_{PK}, Q returns the corresponding public key $\mathrm{PK}_i = (X_i, Y_i)$ to A_{12}; otherwise, it selects a value $c_i \in \{0, 1\}$ randomly, with $\Pr[c_i = 1] = \sigma = \frac{1}{q_{\mathrm{s}} + q_{\mathrm{sk}} + 1}$.

If $c_i = 0$, Q randomly selects $x_i, d_i, h_{i1} \in \mathbb{Z}_q^*$ satisfying $X_i, Y_i \notin L_{\mathrm{PK}}$, and computes $X_i = x_i P$, $Y_i = d_i P - P_{\mathrm{pub}} h_{1i}$. Next, Q adds $(\mathrm{PID}_i, X_i, Y_i, c_i)$ to L_{PK} and returns $\mathrm{PK}_i = (X_i, Y_i)$ to A_{12}. Finally, Q adds $(\mathrm{PID}_i, x_i, d_i)$ and $(\mathrm{PID}_i, X_i, Y_i, h_{1i})$ to L_{SK} and L_1, respectively.

If $c_i = 1$, Q sets $X_i = r_{\mathrm{know3}} P$ and $Y_i = bP$, where $r_{\mathrm{know3}} \in \mathbb{Z}_q^*$ is a random number known by Q, and $X_i, Y_i \in L_{\mathrm{PK}}$. Q adds $(\mathrm{PID}_i, X_i, Y_i, c_i)$ to L_{PK} and returns $\mathrm{PK}_i = (X_i, Y_i)$ to A_{12}.

Decryption queries: Q queries the corresponding tuple $(\mathrm{PID}_B, X_B, Y_B, c_B)$ of PID_B in L_{PK} when Q receives decryption queries from A_{12}.

If $c_B = 0$, Q searches the corresponding private key $\mathrm{SK}_B = (x_B, d_B)$ of PID_B and public key $\mathrm{PK}_i = (X_i, Y_i)$ of PID_i in L_{SK} and L_{PK}, respectively. Next, Q computes $U_i = (x_B + y_B) R_i$ and executes the decryption algorithm to obtain $\mathrm{PID}_i\|m_i\|t$ and $h_{2i} = H_2(\mathrm{PID}_i, X_i, Y_i, R_i, C_i, \varDelta)$. Finally, Q returns $\mathrm{PID}_i\|m_i\|t$ to A_{12}.

If $c_B = 1$, Q searches h_{1i}, h_{2i}, and h_{3i} from L_1, L_2, and L_3 to compute $\mathrm{PID}_i\|m_i\|t = C_i \oplus h_{3i}$, respectively. Q returns $\mathrm{PID}_i\|m_i\|t$ to A_{12} when $s_iP = Y_i + P_{\mathrm{pub}}h_{1i} + X_ih_{2i}$ is satisfied; otherwise, the simulation terminates.

Stage 2: challenge stage

Adversary A_{12} outputs two identities PID_i and PID_B and two equal-length messages m_0 and m_1, where PID_B is a challenger. Next, Q performs public key extraction queries for PID_B to obtain $(\mathrm{PID}_B, X_B, Y_B, c_B)$.

If $c_B = 0$, the simulation terminates.

If $c_B = 1$, Q randomly selects $a, h^*_{1i}, h^*_{2i}, h^*_{3i}, s^*_i, t^* \in \mathbb{Z}^*_q$ to compute $R^*_i = aP$ and makes them satisfy $s^*_iP = Y_i + P_{\mathrm{pub}}h^*_{1i} + X_ih^*_{2i}$, $U^*_i = a(X_B + Y_B + P_{\mathrm{pub}}h^*_{1i}) = (r_{\mathrm{know1}} + r_{\mathrm{know2}} + bh^*_{1i})R^*_i$, and $C^*_i = m_\theta \oplus h^*_{3i}$, $\theta \in \{0, 1\}$. Next, Q returns $\sigma^*_i = (R^*_i, s^*_i, C^*_i, t^*)$ to A_{12}.

Adversary A_{12} performs the above queries for probabilistic polynomial times and outputs guesses on θ', $\theta' \in \{0, 1\}$. If $\theta' = \theta$, Q outputs $abP = U^*_i - R^*_i(r_{\mathrm{know3}} + sh^*_{1i})$ as an effective solution to ECCDHP; otherwise, the difficult problem is not solved.

Probabilistic analysis: Q solves ECCDHP successfully, which means that Q does not stop the simulation and adversary A_{12} breaks through the confidentiality of the proposed scheme with a non-negligible probability ε_{12}. According to Theorem 1, the confidentiality of the scheme can be broken only when adversary A_{12} solves ECCDHP with a non-ignorable probability $\left(1-\frac{q_{\mathrm{sk}}}{2^k}\right)\left(1-\frac{q_3}{2^k}\right)\frac{\varepsilon_{12}}{\mathrm{e}n(q_{\mathrm{s}} + q_{\mathrm{sk}} + 1)}$.

Theorem 3 (Unforgeability under adversary A_{21}) In the random oracle model, the unforgeability of the proposed scheme can be broken if adversary A_{21} can win Game 3 in polynomial time with a non-negligible probability ε_{21} (in the game, A_{21} can make q_{s} signcryption queries and q_{sk} private key extraction queries at most). Then Q can solve ECDLP in polynomial time with at least a non-negligible probability $\left(1 - \frac{q_{\mathrm{sk}}}{2^k}\right)\frac{\varepsilon_{21}}{\mathrm{e}n(q_{\mathrm{s}} + q_{\mathrm{sk}} + 1)}$.

We present the proof of the unforgeability of our certificateless aggregate signcryption scheme based on ECDLP under the random oracle model.

Proof Q is a solver of ECDLP. Given the input (P, aP), its goal is to obtain a when $a \in \mathbb{Z}^*_q$ and is unknown. Q uses adversary A_{21} as a challenger to Game 3. After announcing the start of the game, Q executes the system parameter generation algorithm and sends the public parameters params to A_{21}. Let $P_{\mathrm{pub}} = aP$, and let a act as the system master key. In addition, Q maintains lists L_1, L_2, L_{SK}, and L_{PK}, which are used to track the inquiries of A_{21} about H_1 queries, H_2 queries, private key extraction queries, and public key extraction queries for the oracle model, respectively. Initially, each list is empty.

Stage 1: query stage

Adversary A_{21} performs polynomial bounded-time queries as follows:

Adversary A_{21} performs H_1 queries, H_2 queries, public key extraction queries, private key extraction, and public key replacement queries.

Signature queries: When Q receives a signature query from A_{21}, it looks for $(\mathrm{PID}_i, X_i, Y_i, c_i)$ in L_{PK}.

If $c_i = 1$, the query ends and the simulation terminates.

Otherwise, Q looks for the private key $\mathrm{SK}_i = (x_i, d_i)$ of PID_i in L_{SK} and executes the signcryption algorithm to generate $\sigma_i = (R_i, s_i, C_i, t)$ for A_{21}.

Signature verification queries: Q looks for $(\mathrm{PID}_i, X_i, Y_i, c_i)$ and $(\mathrm{PID}_B, X_B, Y_B, c_B)$ in L_{PK}.

If $c_i = 0$, Q computes $h_{1i} = H_1(\mathrm{PID}_i, X_i, Y_i)$, $h_{2i} = H_2(\mathrm{PID}_i, X_B, Y_B, R_i, C_i, \varDelta)$, and verifies whether $s_iP = Y_i + P_{\mathrm{pub}}h_{1i} + X_ih_{2i}$ is satisfied. If the equation holds, Q executes the decryption algorithm and returns m_i to A_{21}.

If $c_i = 1$, Q looks for h'_{1i} and h'_{2i} in L_1 and L_2, respectively. Next, Q verifies whether $s_iP = Y_i + P_{\mathrm{pub}}h'_{1i} + X_ih'_{2i}$ is satisfied. If it is true, Q returns $\mathrm{PID}_i\|m_i\|t$ to A_{21}; otherwise, the simulation terminates.

If c_i does not exist, it means that the public key has been replaced and Q inquiries L_1 and L_2 to obtain $(\mathrm{PID}_i, X'_i, Y'_i, h'_{1i})$ and $(\mathrm{PID}_i, X'_i, Y'_i, R_i, C_i, \varDelta', h'_{2i})$, respectively. Next, Q verifies whether $s_iP = Y'_i + P_{\mathrm{pub}}h'_{1i} + X_ih'_{2i}$ is satisfied. If it is true, Q returns $\mathrm{PID}_i\|m_i\|t$ to A_{21}; otherwise, the simulation terminates.

Stage 2: challenge stage

Q inquiries L_{PK} for $(\mathrm{PID}_i, X_i, Y_i, c_i)$. If $c_i = 0$, the query ends and the simulation terminates.

Otherwise, $c_i = 1$, Q randomly selects $r_i^*, x_i^*, y_i^*, t^* \in \mathbb{Z}_q^*$ to compute $R_i^* = r_i^* P$, $U_i^* = (x_i^* + d_i^*)R_i^*$, $h_{1i}^* = H_1(\mathrm{PID}_i, X_i, Y_i)$, $h_{2i}^* = H_2(\mathrm{PID}_i, X_B, Y_B, R_i^*, C_i, \Delta^*)$, and $h_{3i}^* = H_3(\mathrm{PID}_i, \mathrm{PID}_B, X_i, Y_i, R_i^*, U_i^*)$. Q outputs n forged signatures $\sigma_i^* = (R_i^*, s_i^*, C_i, t^*)$. Next, Q verifies whether $s_i^* P = Y_i + P_{\text{pub}} h_{1i}^* + X_i h_{2i}^*$ is satisfied. If it is true, σ_i^* is valid; otherwise, the simulation terminates. When Q receives an aggregation query, it computes $S^* = \sum_{i=1}^{n} s_i^*$ to output an aggregate signcryption $\sigma^* = (R_i^*, s^*, C_i, t^*)$. Finally, Q verifies whether $s_i{}^* P = \sum_{i=1}^{n} Y_i + P_{\text{pub}} \sum_{i=1}^{n} h_{1i}^* + \sum_{i=1}^{n} X_i h_{2i}^*$ is satisfied. When it is true, the aggregation signcryption is forged successfully, and Q outputs the solution to ECDLP, i.e., $a = \frac{1}{h_{1i}^*}\left[S^* - \sum_{i=1}^{n}(r_{\text{know1}} + r_{\text{know2}} h_{2i}^*)\right]$; otherwise, ECDLP cannot be solved.

Probabilistic analysis: Q solves ECDLP successfully, which means that it does not stop the simulation and adversary A_{21} breaks through the unforgeability of this scheme with a non-negligible probability ε_{21}. Q challenges successfully only with the following events:

ε_1: At least one user does not ask for private key generation with a probability of $\Pr[\varepsilon_1] = \frac{1}{n}\left(1 - \frac{q_{\text{sk}}}{2^k}\right)$.

ε_2: Q does not terminate the query phase with a probability of $\Pr[\varepsilon_2] = (1-\delta)^{q_s + q_{\text{sk}} + 1}$. When $q_s + q_{\text{sk}}$ is large enough, $(1-\delta)^{q_s + q_{\text{sk}} + 1}$ tends to e^{-1}.

ε_3: Q does not terminate the challenge phase with the probability of $\Pr[\varepsilon_3] = \delta$.

Therefore, the unforgeability of this scheme can be broken only if adversary A_{21} solves ECDLP with a non-negligible probability $\left(1 - \frac{q_{\text{sk}}}{2^k}\right) \cdot \frac{\varepsilon_{21}}{\mathrm{e}n(q_s + q_{\text{sk}} + 1)}$.

Theorem 4 (Unforgeability under adversary A_{22}) In the random oracle model, the unforgeability of the proposed scheme can be broken if adversary A_{22} can win Game 4 in polynomial time with a non-negligible probability ε_{22} (in the game, A_{22} can make q_s signcryption queries and q_{sk} private key extraction queries at most). Then, Q can solve ECDLP in polynomial time with at least a non-negligible probability $\left(1 - \frac{q_{\text{sk}}}{2^k}\right)\frac{\varepsilon_{22}}{\mathrm{e}n(q_s + q_{\text{sk}} + 1)}$.

Proof Q is a solver of ECDLP. Given the input (P, aP), its goal is to obtain a when $a \in \mathbb{Z}_q^*$ and is unknown. Q uses adversary A_{22} as a challenger of the game. After announcing the start of the game, Q executes the system parameter generation algorithm and sends the public parameters params and s to A_{22}. In addition, Q maintains lists L_1, L_2, L_{SK}, and L_{PK} to track the inquiries of A_{22}, including H_1 queries, H_2 queries, private key extraction queries, and public key extraction queries, respectively. Initially, each list is empty.

Stage 1: query stage

Adversary A_{22} performs polynomial bounded-time queries as follows:

Adversary A_{22} performs H_1 queries, H_2 queries, public key extraction queries, private key extraction queries in Theorem 2, and public key replacement queries in Theorem 3.

Signature verification inquiry: Q looks for $(\mathrm{PID}_i, X_i, Y_i, c_i)$ and $(\mathrm{PID}_B, X_B, Y_B, c_B)$ in L_{PK}.

If $c_i = 0$, Q computes $h_{1i} = H_1(\mathrm{PID}_i, X_i, Y_i)$, $h_{2i} = H_2(\mathrm{PID}_i, X_B, Y_B, R_i, C_i, \Delta)$, and verifies whether $s_i P = Y_i + P_{\text{pub}} h_{1i} + X_i h_{2i}$ is satisfied. If the equation holds, Q executes the decryption algorithm and returns $\mathrm{PID}_i \| m_i \| t$ to A_{22}; otherwise, the simulation terminates.

If $c_i = 1$, Q looks for h'_{1i} and h'_{2i} in L_1 and L_2, respectively. Next, Q verifies whether $s_i P = Y_i + P_{\text{pub}} h'_{1i} + X_i h'_{2i}$ is satisfied. If it is true, Q returns $\mathrm{PID}_i \| m_i \| t$ to A_{22}; otherwise, the simulation terminates.

If c_i does not exist, the public key has been replaced, and Q queries L_1 and L_2 to obtain $(\mathrm{PID}_i, X_i, Y_i, h'_{1i})$ and $(\mathrm{PID}_i, X_i, Y_i, R_i, C_i, \Delta', h'_{2i})$, respectively. Next, Q verifies whether $s_i P = Y_i + P_{\text{pub}} h'_{1i} + X_i h'_{2i}$ is correct. If it is true, Q returns $\mathrm{PID}_i \| m_i \| t$ to A_{22}; otherwise, the simulation terminates.

Stage 2: challenge stage

Q queries L_{PK} for $(\mathrm{PID}_i, X_i, Y_i, c_i)$. If $c_i = 0$, the query ends and the simulation terminates; if $c_i = 1$, Q selects $r_i^*, x_i^*, y_i^*, t^* \in \mathbb{Z}_q^*$ randomly to compute $R_i^* = r_i^* P$, $h_{1i}^* = H_1(\mathrm{PID}_i, X_i, Y_i)$, $h_{2i}^* = H_2(\mathrm{PID}_i, X_B, Y_B, R_i^*, C_i, \Delta^*)$, and then outputs n forged signcryptions $\sigma_i^* = (R_i^*, s_i^*, C_i, t^*)$. Next, Q verifies whether $s_i^* P = Y_i + P_{\text{pub}} h_{1i}^* + X_i h_{2i}^*$ is satisfied. If it is true, σ_i^* is valid; otherwise, the simulation terminates. When Q receives an aggregation query, it computes

$S^* = \sum_{i=1}^{n} s_i^*$ to output aggregation signcryption $\sigma^* = (R_i^*, s_i^*, C_i, t^*)$. Finally, Q verifies whether $s_i^* P = \sum_{i=1}^{n} Y_i + P_{\text{pub}} \sum_{i=1}^{n} h_{1i}^* + \sum_{i=1}^{n} X_i h_{2i}^*$ is satisfied. If it is true, σ^* is forged successfully. Then, Q outputs the solution to ECDLP, i.e., $a = S^* - \sum_{i=1}^{n} (s h_{1i}^* + r_{\text{know3}} h_{2i}^*)$. Otherwise, ECDLP cannot be solved.

Probabilistic analysis: Q solves ECDLP successfully, which means that it does not stop the simulation and adversary A_{22} breaks through the unforgeability of the proposed scheme with a non-negligible probability ε_{22}. According to Theorem 4, the unforgeability of the proposed scheme can be broken only if adversary A_{22} solves ECDLP with a non-negligible probability $\left(1 - \frac{q_{\text{sk}}}{2^k}\right) \frac{\varepsilon_{22}}{\mathrm{e}n(q_{\text{s}} + q_{\text{sk}} + 1)}$.

5 Scheme analysis

5.1 Safety characteristic analysis

1. Eliminating key hosting

This scheme adopts the certificateless technology. When a user provides his/her identity information, KGC takes the master key s and the user's identity information as parameters to generate a partial private key for the user. The secret value selected by the user and the partial key from KGC form the whole key for the user. Consequently, the problem of key hosting is solved.

2. Resistance to replay attack

When a third party intercepts a request packet of the encryption processing sent by the client to the server, it cannot decrypt the data acquired, but can repeatedly send the package to the server for repetitive request operations (Ramanan et al., 2021). A server without a replay attack prevention function may increase pressure and lead to data disorder. Therefore, our scheme adds a time stamp in the signcryption to effectively prevent replay attacks.

3. Relief distributed denial-of-service (DDoS) attacks

The structure of PRDF is based on cloud-fog cooperation mode. By introducing fog layer devices between cloud and the terminal devices, the distributed computing and storage of fog computing can alleviate problems like the large transmission distance of traditional cloud computing and vulnerability to DDoS attacks.

Table 3 shows a comparison of safety characteristics of different schemes for the PIoT.

5.2 Performance analysis

When comparing the computational efficiency of signcryption schemes, assuming that n users participate in the signcryption and that the computational overhead depends mainly on the following operations: E_{e} (exponent arithmetic), E_{p} (bilinear pairing operation), E_{m} (multiplication of points defined on elliptic curves on group $\mathbb{G}$), and E_{a} (addition of points defined on elliptic curves). The computational overhead of E_{p} is more than 10 times that of E_{m}. A huge number of PIoT users lead to a large amount of electricity data to aggregate. Therefore, our work does not use E_{e} or E_{p}. The computational overhead depends mainly on E_{m} and E_{a}.

To compare the performance of different schemes quantitatively, the execution time of E_{p},

Table 3 Comparison of safety characteristics

Scheme	Confidentiality	Unforgeability	Relief DDoS attacks	Resistance to replay attack	Eliminating key hosting
Yu CM et al. (2014)'s	×	√	×	√	×
Wang L (2019)'s	×	√	×	√	×
Wang XD et al. (2021)'s	×	√	×	×	×
Chen (2016)'s	√	√	×	√	√
Sui and de Meer (2020)'s	√	√	×	√	×
Xie and Li (2020)'s	√	√	×	√	√
Ours	√	√	√	√	√

E_m, and E_a is obtained through experiments. The execution time of each operation is shown in Table 4. These operations are executed in a laptop with the Intel® Core™ i7-6700HQ 2.59 GHz processor, 8 GB RAM, and Windows 10 operating system. The elliptic curve is $y^2 = x^3 + ax + b \bmod p$, where p is 160 bits.

Table 4 Execution time

Symbol	Operation	Time (ms)
E_p	Bilinear pairing	4.1486
E_m	Multiplication of points	0.4738
E_a	Addition of points	0.0021

The performance of existing methods and the length of ciphertexts are shown in Table 5. The length of ciphertexts represents the communication overhead of each scheme. To analyze the performance of every scheme under the same condition, we assume that the length of plaintexts is the same as l_m. The communication overhead in the literature (Zhang SM et al., 2018; Cui et al., 2019; Nkenyereye et al., 2019; Yu HF and Ren, 2022) is the same in this study; it is $l_m + |\mathbb{G}| + |\mathbb{Z}_q^*| = 640$ bits. The ciphertext of Li C and Qi (2020) contained aggregated broadcast values and Kim et al. (2020) used the generating elements of two groups to calculate the signcryption of User$_i$. Therefore, their communication cost is the highest, $l_m + 2|\mathbb{G}| + |\mathbb{Z}_q^*| = 960$ bits.

The signature and aggregation cost of our work is lower than that of Cui et al. (2019)'s scheme, but the decryption and verification efficienies are higher. The decryption and verification cost is lower than that in Li C and Qi (2020), but the signcryption and aggregation efficiencies are higher. Moreover, the length of ciphertext in our work is shorter than that in Li C and Qi (2020). The efficiency comparison is shown in Table 5 (the ciphertext length in each scheme includes the plaintext length l_m, so we give only the parts other than l_m).

In the simulations, n (the number of users belonging to the same FN) is set to 1000 in Fig. 4. Fig. 5 shows the comparison of the calculation cost of each scheme when n is different. As can be seen from Figs. 4 and 5, our scheme has more advantages compared with the other schemes.

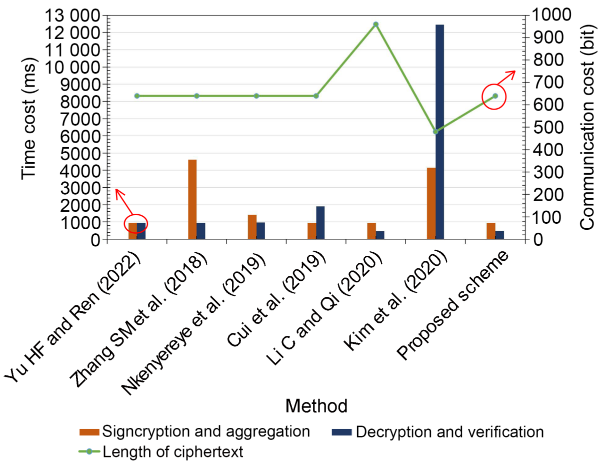

Fig. 4 Comparison of calculation and communication costs of each scheme

6 Conclusions

To solve the privacy protection problem of users' power data in the PIoT and provide users with exclusive power services, in this paper we proposed PRDF with a privacy protection function. PRDF is effective for facilities in the PIoT. Theoretical analysis

Table 5 Efficiency comparison among certificateless aggregate signcryption schemes

Scheme	Time cost (ms)		Communication cost (bit)
	Signcryption aggregation	Decryption verification	Length of ciphertexts
Yu HF and Ren (2022)'s	$(2n+1)E_m + 2nE_a$	$(2n+2)E_m + 3E_a$	$\|\mathbb{G}\|+\|\mathbb{Z}_q^*\|$
Zhang SM et al. (2018)'s	$nE_p + nE_m + nE_a$	$3E_p + 2nE_m + nE_a$	$\|\mathbb{G}\|+\|\mathbb{Z}_q^*\|$
Nkenyereye et al. (2019)'s	$(3n+1)E_m + nE_a$	$4E_p + 2nE_m$	$\|\mathbb{G}\|+\|\mathbb{Z}_q^*\|$
Cui et al. (2019)'s	$2nE_m$	$(4n+1)E_m + 3nE_a$	$\|\mathbb{G}\|+\|\mathbb{Z}_q^*\|$
Li C and Qi (2020)'s	$(2n+1)E_m + (2n+2)E_a$	$nE_m + nE_a$	$2\|\mathbb{G}\|+\|\mathbb{Z}_q^*\|$
Kim et al. (2020)'s	nE_p	$(3n+2)E_p$	$2\|G\|+\|\mathbb{Z}_q^*\|$
Ours	$(2n+1)E_m + 2E_a$	$(n+2)E_m + (2n+2)E_a$	$\|\mathbb{G}\|+\|\mathbb{Z}_q^*\|$

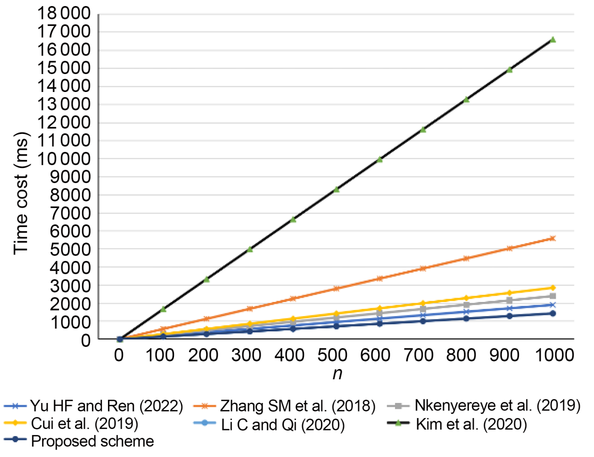

Fig. 5 Comparison of calculation time cost of each scheme (0≤*n*≤1000)

and simulation results showed that our scheme is more efficient and has more security characteristics.

Contributors

Shuanggen LIU designed the research. Shuangzi ZHENG and Wenbo ZHANG processed the data. Shuangzi ZHENG drafted and organized the paper. Shuangzi ZHENG and Runsheng FU revised and finalized the paper.

Compliance with ethics guidelines

Shuanggen LIU, Shuangzi ZHENG, Wenbo ZHANG, and Runsheng FU declare that they have no conflict of interest.

References

Cai JY, 2021. Power big data analysis technology and application analysis supported by cloud computing technology. *Electron World*, (6):79-80 (in Chinese). https://doi.org/10.19353/j.cnki.dzsj.2021.06.038

Chen JQ, 2016. The Research on Smart Grid Privacy Protection Method. MS Thesis, Hunan University, Changsha, China (in Chinese).

Cui MM, Han DZ, Wang J, 2019. An efficient and safe road condition monitoring authentication scheme based on fog computing. *IEEE Internet Things J*, 6(5):9076-9084. https://doi.org/10.1109/JIOT.2019.2927497

Guo C, Jiang XR, Choo KKR, et al., 2020. Lightweight privacy preserving data aggregation with batch verification for smart grid. *Future Gener Comput Syst*, 112:512-523. https://doi.org/10.1016/j.future.2020.06.001

Jia WJ, Zhou XJ, 2018. Concepts, issues, and applications of fog computing. *J Commun*, 39(5):153-165 (in Chinese). https://doi.org/10.11959/j.issn.1000-436x.2018086

Jin X, 2021. Discussion on problems in the application of smart grid and big data technology. *Vadose Zone J*, 52(8): 183-184 (in Chinese).

Kim TH, Kumar G, Saha R, et al., 2020. CASCF: certificateless aggregated signcryption framework for Internet-of-Things infrastructure. *IEEE Access*, 8:94748-94756. https://doi.org/10.1109/ACCESS.2020.2995443

Li C, Qi ZH, 2020. An efficient and safe certificateless signcryption scheme. *Comput Technol Dev*, 30(10):117-122 (in Chinese). https://doi.org/10.3969/j.issn.1673-629X.2020.10.022

Li HJ, Gao Q, 2021. Overview of privacy protection technologies for smart meters using rechargeable batteries. *J Shanghai Univ Electr Power*, 37(1):23-26, 43 (in Chinese). https://doi.org/10.3969/j.issn.2096-8299.2021.01.005

Lyu LJ, Nandakumar K, Rubinstein B, et al., 2018. PPFA: privacy preserving fog-enabled aggregation in smart grid. *IEEE Trans Ind Inform*, 14(8):3733-3744. https://doi.org/10.1109/TII.2018.2803782.

Ma B, Yuan L, Liu WZ, et al., 2019. Distribution mechanism for smart grid based on cloud-fog computing. *Electr Meas Instrum*, 56(24):67-72 (in Chinese). https://doi.org/10.19753/j.issn1001-1390.2019.024.011

Ma JJ, Zhang ZQ, Cao SZ, et al., 2021. Distributed attribute-based encryption scheme based on fog nodes. *Comput Eng*, 47(6):38-43 (in Chinese). https://doi.org/10.19678/j.issn.1000-3428.0060063

Nkenyereye L, Liu CH, Song JS, 2019. Towards secure and privacy preserving collision avoidance system in 5G fog based Internet of Vehicles. *Future Gener Comput Syst*, 95:488-499. https://doi.org/10.1016/j.future.2018.12.031

Ramanan P, Li D, Gebraeel N, 2021. Blockchain-based decentralized replay attack detection for large-scale power systems. *IEEE Trans Syst Man Cybern Syst*, 52(8):4727-4739. https://doi.org/10.1109/TSMC.2021.3104087

Shen H, Liu YJ, Xia Z, et al., 2020. An efficient aggregation scheme resisting on malicious data mining attacks for smart grid. *Inform Sci*, 526:289-300. https://doi.org/10.1016/j.ins.2020.03.107

Sui ZY, de Meer H, 2020. An efficient signcryption protocol for hop-by-hop data aggregations in smart grids. *IEEE J Sel Areas Commun*, 38(1):132-140. https://doi.org/10.1109/JSAC.2019.2951965

Ul Hassan M, Rehmani MH, Kotagiri R, et al., 2019. Differential privacy for renewable energy resources based smart metering. *J Parall Distrib Comput*, 131:69-80. https://doi.org/10.1016/j.jpdc.2019.04.012

Wang L, 2019. Research on Provable Secure Aggregate Signature Scheme and Its Application. MS Thesis, East China Jiaotong University, Nanchang, China (in Chinese). https://doi.org/10.27147/d.cnki.ghdju.2019.000367

Wang QY, Chen J, Zhuang LS, 2020. Batch verification of linkable ring signature in smart grid. *J Cryptol Res*, 7(5): 616-627 (in Chinese). https://doi.org/10.13868/j.cnki.jcr.000394

Wang XD, Liu YN, Choo KKR, 2021. Fault-tolerant multi-subset aggregation scheme for smart grid. *IEEE Trans Ind Inform*, 17(6):4065-4072. https://doi.org/10.1109/TII.2020.3014401

Xia ZQ, Zhang YC, Gu K, et al., 2022. Secure multidimensional and multi-angle electricity data aggregation scheme for fog computing-based smart metering system. *IEEE Trans Green Commun Netw*, 6(1):313-328. https://doi.org/10.1109/TGCN.2021.3122793

Xie GM, Li SL, 2020. Smart grid data privacy preserving scheme based on noise and aggregation signcryption. *Mod Comput*, 26(10):18-22 (in Chinese). https://doi.org/10.3969/j.issn.1007-1423.2020.10.004

Xu JW, Ota K, Dong MX, et al., 2018. SIoTFog: Byzantine-resilient IoT fog networking. *Front Inform Technol Electron Eng*, 19(12):1546-1557. https://doi.org/10.1631/FITEE.1800519

Yu CM, Chen CY, Kuo SY, et al., 2014. Privacy-preserving power request in smart grid networks. *IEEE Syst J*, 8(2): 441-449. https://doi.org/10.1109/JSYST.2013.2260680

Yu HF, Ren RT, 2022. Certificateless elliptic curve aggregate signcryption scheme. *IEEE Syst J*, 16(2):2347-2354. https://doi.org/10.1109/JSYST.2021.3096531

Zhang LY, 2021. Research on smart grid dispatching platform based on cloud computing. *Power Syst Big Data*, 24(2): 34-40 (in Chinese). https://doi.org/10.19317/j.cnki.1008-083x.2021.02.005

Zhang SM, Zhao YQ, Wang BY, 2018. Certificateless ring signcryption scheme for preserving user privacy in smart grid. *Autom Electr Power Syst*, 42(3):118-123, 135 (in Chinese). https://doi.org/10.7500/AEPS20170817006