Hindawi
Wireless Communications and Mobile Computing
Volume 2023, Article ID 9884050, 14 pages
https://doi.org/10.1155/2023/9884050

Research Article

An Efficient Pairing-Free Certificateless Signature Scheme with KGC Trust Level 3 for Wireless Sensor Network

Hong Zhao,[1,2] **Xinyu Zhang**,[2] **Zhaobin Li**,[2] **and Zhanzhen Wei**[2]

[1]*School of Cyber Science and Technology, University of Science and Technology of China, Heifei 230026, China*
[2]*Department of Electronic and Communication Engineering, Beijing Electronic Science and Technology Institute, Beijing 100070, China*

Correspondence should be addressed to Zhanzhen Wei; wzz@besti.edu.cn

Received 14 June 2022; Revised 3 January 2023; Accepted 17 March 2023; Published 9 May 2023

Academic Editor: Ghufran Ahmed

With the widespread adoption of wireless sensor networks (WSN), the security of the WSN has been a wide concern. Certificateless signature eliminates the certificate management problem and key escrow problem and is considered a feasible solution to solve the data integrity and authentication of WSN. Recently, Thumbur et al. proposed an efficient pairing-free certificateless signature scheme, and Xu et al. pointed out that their scheme is not resistant to signature forgery attacks and proposed an improved scheme. Based on the trust hierarchy defined by Girault, we find that Xu et al.'s scheme is still only able to achieve security under KGC trust level 2. Moreover, Thumbur et al.'s scheme uses the Schnorr signature algorithm form, which makes it favorable for scaling, while Xu et al.'s scheme breaks this advantage. Therefore, we propose a pairing-free certificateless scheme capable of reaching KGC trust level 3, still using the Schnorr signature algorithm form, and prove the security of the new scheme under the random oracle model. The final efficiency analysis shows that the new scheme has shorter public key length and higher computational efficiency.

1. Introduction

In recent years, wireless sensor network (WSN) technology has been developed tremendously, which has triggered widespread interest in both academia and industry. WSN is a self-organized multihop network of a large number of sensor nodes with features such as flexibility, fault tolerance, high perception ability, and rapid layout. These features of WSN determine its wide range of applications, such as environmental monitoring, agriculture, military, Smart Grid [1], and medical [2, 3]. As shown in Figure 1, the WSN system consists of aggregation node (i.e., sink node), sensor nodes, and management node. In practice, sensor nodes are arbitrarily deployed in the monitored area by manual placement or drone dispersal, forming a WSN through wireless self-organization, in which each node has the function of a router and can locate and restore connections. When the WSN is in operation, the sensor nodes collect the required information and transmit it to the sink node in the form of single-hop or multihop. The sink node performs preliminary data processing and information fusion and then transmits it to the users through satellite channels or wired network connections [4]. A wireless communication channel adopted by WSN is easily monitored by attackers, leading to information leakage or tampering. Sensor nodes in WSN are often distributed in unattended, harsh, or even hostile open environments, and the nodes are easily captured or physically controlled by attackers. Compared with traditional internet networks, WSN faces more complex and diverse security threats, so existing cybersecurity mechanisms are not fully applicable to WSN. Since most WSN devices are usually based on small embedded chips with limited computing and storage capabilities and most of them adopt wireless technologies such as Bluetooth and ZigBee for communication, the communication bandwidth of WSN is limited by the spectrum resources of wireless communication. Therefore, how to ensure the communication security of WSN is a challenge in the case of limited computing, storage, and communication capabilities.

Digital signature technology can provide the authentication, data integrity, and nonrepudiation functions required by WSN. In digital signature schemes, the problem of how

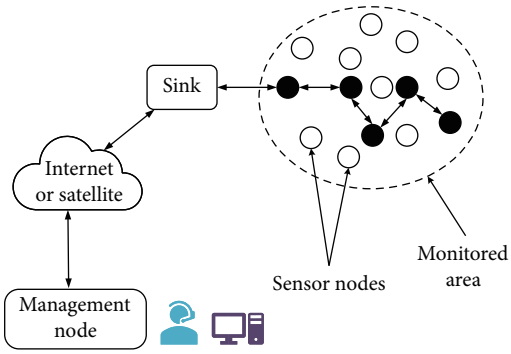

Figure 1: Wireless sensor network model.

to bind the user's identity to the user's public key needs to be solved; otherwise, the risk of man-in-the-middle public key replacement attack is faced. Currently, there are three main solutions. The first is public key infrastructure (PKI) cryptosystem, which faces complex certificate management problems. The second is identity-based public key cryptography (IBC) system, which simplifies the complexity of certificate management but has the problem of key escrow. The third is certificateless public key cryptography (CL-PKC) system, where the user's private key consists of a partial private key (PPK) generated by the key generation center (KGC) and the secret value generated by the user together, and there is no certificate in the system, so there is no certificate management problem, and the KGC cannot calculate the complete private key of the user, thus solving the key escrow problem. Therefore, this paper focuses on certificateless signature (CLS) schemes applied to WSN.

2. Related Work

In 2003, Al-Riyami and Paterson [5] proposed the first CL-PKC scheme, and subsequently, Yum and Lee [6] proposed a generic construction for CLS in 2004. After that, many CLS schemes are devised [7–11]. However, these schemes were constructed based on pairing operations in ECC and map-to-point hash functions, which require high computing resources and are not suitable for implementation on resource-constrained WSN devices. In 2011, He et al. [12] proposed the first pairing-free certificateless signature (PF-CLS) scheme, which did not use pairing operations and greatly improved the computational efficiency. Later, many PF-CLS schemes [13–18] were proposed for resource-constrained environments.

In the security model in [5], two types of adversaries are defined. The Type-I adversary represents malicious signer, which does not know the system master key but can replace the signer's public key at will, and the Type-II adversary represents malicious KGC which knows the system master key but cannot replace the signer's public key. In 2006, Au et al. [19] proposed a new kind of attack: malicious-but-passive KGC attack. This attack assumes that the KGC is already malicious at the beginning of the system setup phase. This KGC maliciously generates a specific system master public/secret keypair for a specific user, and then when that user publishes his public key, the KGC is able to calculate the user's secret value by the user's public key. In 2007, Huang et al. [20] classified Type-I/II adversaries into three types: normal, strong, and super Type-I/II adversaries, but all of them were based on the capability limits of the two types of adversaries defined in [5]. Once KGC can replace the user's public key, it can impersonate any user, which leads to the fact that the user must trust the KGC completely. According to Girault's [21] definition of trust hierarchy for authority (KGC is the authority of CL-PKC system) in the public key cryptosystem:

(i) Trust level 1 implies that the trusted authority can compute the private keys of all the users in the system

(ii) Trust level 2 implies that authority cannot compute the private keys of individual users; however, it can still generate false guarantees to impersonate any user in the system, without being implicated

(iii) Trust level 3 implies that the authority cannot compute the user's private key, and it will be implicated if the KGC generates false guarantees

Many CL-PKC systems just achieve trust level 2, whereas the PKI technology attains trust level 3, and the IBC technology only achieves trust level 1. Al-Riyami and Paterson [5] pointed out that the CL-PKC system with trust level 2 could be upgraded to trust level 3 through binding technology, i.e., the public key corresponding to the user secret value is bound to the user ID. But the relevant security models and proofs were not further elaborated. In 2011, Yang and Tan [22] introduced a new binding technique, proposed the notion of key-dependent certificateless signature (KD-CLS), and directly defined KGC trust level 3 security in the security model described in [5]. The security definition of Yang and Tan [22] did not define a new type of adversary, but only placed different restrictions on the capabilities of Type-I/II adversaries defined by Al-Riyami and Paterson [5]. In 2017, Li et al. [23], based on the literature [22], further pointed out that the scheme using the binding technology described in [5] can also be proved secure. In addition, in 2007, Hu et al. [24] defined the KGC trust level 3 security model for CLS separately from the definition in [5]. In this definition, a new type of adversary is introduced, and such an adversary is required to be able to forge the user's legitimate public-private key pair. This independent security definition had since been further elaborated and applied by Chen et al. [25] in 2015 and Tseng et al. [26] in 2019. In 2021, Rastegari and Susilo [27] updated the victory condition of this independent security definition to be that the legitimate user whose signature is forged cannot repudiate the forged signature. The security definition in this paper is mainly based on literature [26] and literature [27].

At present, many researchers on PF-CLS schemes ignore the KGC trust level issue and focus only on how to resist the attacks of Type-I/II adversaries with KGC trust level 2. For example, Yeh et al. [13] proposed a PF-CLS scheme for the IoT in 2017. In 2018, Jia et al. [14] pointed out that Yeh

et al.'s scheme could not resist the public key replacement attack of Type-I adversary. In 2020, Du et al. [15] further pointed out that Jia et al.'s scheme cannot resist the public key replacement attack of Type-I adversary, and in 2022, Xiang et al. [18] further stated that Jia et al.'s scheme could not resist Type-II adversary attacks. In 2020, Thumbur et al. [16] proposed a PF-CLS scheme for resource-constrained devices. In 2021, [17] proved that Thumbur et al.'s scheme cannot resist the attacks from Type-I adversaries and proposed an improved scheme. In this paper, we analyze the scheme of Xu et al. [17] as an example, which cannot support the KGC trust level 3, and so do the schemes in [13–16, 18].

In addition, most of the PF-CLS schemes mentioned above adopted custom signature algorithms, which was less scalable. The Schnorr signature algorithm [28] has been rigorously proven to be secure [29] and has linear characteristics that make it easy to aggregate and provide scalability guarantees that have been accepted by applications such as Bitcoin. In Thumbur et al.'s scheme [16], the user's complete private key is a sum of PPK generated by KGC and the secret value generated by the user, and the signature algorithm takes the form of a Schnorr signature algorithm, making it easier to further extend to applications such as aggregate signature and multisignatures. However, Xu et al.'s scheme [17] uses a custom signature algorithm, which failed to maintain this advantage and reduced the efficiency of the original scheme. Different from [17], we propose another improved scheme for Thumbur et al.'s scheme [16]. The new scheme still uses the Schnorr signature algorithm with no reduction in computational efficiency, but with a shorter public key length and support KGC trust level 3 security.

2.1. Our Contribution. The main contributions of this paper are the following:

(i) We first analyzed Xu et al.'s scheme and prove their scheme only achieve KGC trust level 2

(ii) We propose a new PF-CLS scheme with KGC trust level 3 and prove the security of the scheme in the random oracle model

(iii) Our scheme uses the Schnorr signature algorithm, which makes the scheme more scalable

(iv) Our scheme has a shorter public key size, and the efficiency analysis shows that our scheme has a lower computational cost

2.2. Paper Organization. The remainder of this paper is organized as follows. We present the relevant PF-CLS works in Section 1. Then, we introduce some preliminaries and security model for PF-CLS scheme in Section 2. A review of Xu et al.'s scheme and security analysis are presented in Sections 4 and 5. Section 6 proposes our PF-CLS scheme for WSN environments. Section 7 provides the correctness proof and security analysis of our scheme. Section 8 gives a comparative analysis of the proposed scheme, with Section 9 giving the paper's conclusions.

Table 1: List of the symbols.

Symbol	Meaning
p, q	Prime number, considerably large in size
G	Additive cyclic group of elliptic curve points of order q
P	Generator of the group G
s_{kgc}	Master secret key of the KGC
P_{pub}	Public key of KGC
params	Public parameters
ID_i	Identity of the user i
x_i	Secret value of the user i
D_i	Partial-private-key of the user i
P_i	Public key of the user i
s_i	Private key of the user i
m	Message
σ_i	Signature of the user i on message m
ID_i^*	Identity chosen by the adversary
P_i^*	Public key replaced by the adversary
D_i^*	Partial-private-key corresponding to P_i^*
m_i^*	Message chosen by the adversary
σ_i^*	Forged signature

3. Preliminaries

In this section, we briefly review some preliminary knowledge, including the definition of elliptic curve discrete logarithm problem, syntax of PF-CLS scheme, and security model for PF-CLS.

To enhance readability, the list of symbols is shown in Table 1.

3.1. Elliptic Curve Discrete Logarithm Problem. F_p denotes a prime finite field, and p is a large prime number. $E(F_p)$ denotes an elliptic curve defined over a finite field F_p by the equation $y^2 = x^3 + ax + b \bmod p$, where $a, b \in F_p$ and 4 $a^3 + 27b^2 \neq 0 \bmod p$. All points on $E(F_p)$ and the infinity point O form a cyclic group G under the operation of point addition $T = U + V$ for $U, V \in G$ defined based on the basis of a chord-and-tangent rule.

Assume G is an additive cyclic group of elliptic curve with order q, where q is a large prime number. $P \in G$ is a generator. Let $x \in Z_q^*$, and scalar multiplication is defined by the equation: $xP = (P + P + \cdots + P)(x \text{ times})$. Given a point $Q \in G$, the elliptic curve discrete logarithm problem (ECDLP) is to find a integer $x \in Z_q^*$ in polynomial time such that $Q = xP$ with nonnegligible probability.

3.2. Syntax of PF-CLS Scheme. As defined by Al-Riyami and Paterson [5], a CLS scheme consists of three entities: a KGC, a signer and a verifier, and seven polynomial-time algorithms. On the basis of [5], He et al. [12] further gave the

definition of the PF-CLS scheme. Following the works [5, 12], we present the syntax definition as follows.

(i) Setup: this algorithm is operated by the KGC. On inputting a security parameter k, it outputs a master public/secret key pair $(P_{\text{pub}}, s_{\text{kgc}})$, the master public key P_{pub} together with other elliptic curve related parameters to form the public parameter params, KGC publishes params and keeps s_{kgc} secretly

(ii) Set-Secret-Value: this algorithm is operated by the user i. On inputting params, it returns x_i as user i 's secret value and X_i as user i's public value

(iii) Partial-Private-Key-Extract: this algorithm is performed by the KGC. On inputting params, system master key s_{kgc}, user's identity ID_i, and public value X_i, it returns D_i as PPK to the user through secure channel

(iv) Set-Private-Key: this algorithm is performed by the user i. On inputting params, D_i and x_i, it returns the user's private key s_i

(v) Set-Public-Key: this algorithm is run by the user i. On inputting params, D_i, it returns the user's public key P_i

(vi) Sign: this algorithm is operated by a user (signer). On inputting *params*, message m, user's identity I D_i and private key s_i, it outputs σ_i as a signature

(vii) Verify: this algorithm is performed by a verifier. Given params, m, σ_i, ID_i, and P_i, it returns "Accept" if σ_i is valid; "Reject" otherwise

Different from the definitions of schemes such as [16, 17], in our definition, Set-Secret-Value is executed before Partial-Private-Key-Extract. The reason is that in order to achieve KGC trust level 3, the Partial-Private-Key-Extract algorithm requires the output X_i of the Set-Secret-Value as input. The Set-Secret-Value algorithm takes the system public parameters params and the user's identity ID_i as input and does not depend on the output of the Partial-Private-Key-Extract algorithm. Therefore, it is feasible for Set-Secret-Value to be executed before Partial-Private-Key-Extract.

3.3. Security Model for PF-CLS. Based on the definition of the security model in [27], if the KGC trust level 3 security needs to be achieved, there exist three types of adversaries: $\mathcal{A}_{\text{I}}$, $\mathcal{A}_{\text{II}}$, and $\mathcal{A}_{\text{III}}$. We utilize the following three games to signify that a CLS scheme is existentially unforgeable against adaptively chosen message and identity attacks (EUF-CMA) against three types of adversaries: $\mathcal{A}_{\text{I}}$, $\mathcal{A}_{\text{II}}$, and $\mathcal{A}_{\text{III}}$.

3.3.1. Game I. The game is executed between a challenger $\mathcal{C}_{\text{I}}$ and a Type-I adversary $\mathcal{A}_{\text{I}}$. And the game proceeds in three phases:

(i) *Setup*. $\mathcal{C}_{\text{I}}$ operates the Setup algorithm to generate a system master key s_{kgc} and the system public parameters params. $\mathcal{C}_{\text{I}}$ sends the params to $\mathcal{A}_{\text{I}}$ and keeps s_{kgc} secretly

(ii) *Query*. $\mathcal{A}_{\text{I}}$ is allowed to issue polynomial queries to the challenger $\mathcal{C}_{\text{I}}$

(a) *Create-User*: upon receiving such a query on ID_i, $\mathcal{C}_I$ calls out a list L_{Cuser} to check if the identity has been created. If yes, $\mathcal{C}_{\text{I}}$ outputs P_i as the public key to $\mathcal{A}_{\text{I}}$. Otherwise, $\mathcal{C}_{\text{I}}$ executes algorithm Set-Secret-Value, Partial-Private-Key-Extract, Set-Private-Key, and Set-Public-Key to produce x_i, D_i, and P_i, respectively. Next, $\mathcal{C}_{\text{I}}$ adds the tuple $(\text{ID}_i, D_i, x_i, P_i)$ to the list L_{Cuser} and outputs P_i to $\mathcal{A}_I$

(Note: we suppose that the Create-User query always precedes other oracle queries)

(b) *Extract-Partial-Private-Key:* when this query on ID_i, $\mathcal{C}_I$ finds the relevant record (ID_i, D_i, x_i, P_i) from L_{Cuser} and returns D_i to $\mathcal{A}_{\text{I}}$

(c) *Extract-Secret-Value*: upon receiving such a query on ID_i, $\mathcal{C}_I$ finds the relevant record $(\text{ID}_i, D_i, x_i, P_i)$ from L_{Cuser} and returns x_i to $\mathcal{A}_{\text{I}}$

(d) *Replace-Public-Key:* given an identity ID_i and a public key P_i^*, this oracle allows $\mathcal{A}_{\text{I}}$ to replace the original public key P_i with P_i^*. Next, $\mathcal{C}_{\text{I}}$ update the tuple $(ID_i,\perp,\perp,P_i)$ to list L_{Cuser}

(e) *Sign:* upon receiving a query on a message m_i, an identity ID_i, and the current public key P_i. $\mathcal{C}_I$ executes the Sign algorithm to generate a valid signature σ_i and outputs it to $\mathcal{A}_{\text{I}}$

(iii) *Forgery*. At last, $\mathcal{A}_{\text{I}}$ outputs a rightful message/signature tuple (m_i^*, σ_i^*) for ID_i^* with P_i^*. The $\mathcal{A}_{\text{I}}$ wins the game if the following three conditions are satisfied:

(1) $\mathcal{A}_{\text{I}}$ outputed a rightful message-signature pair (m_i^*, σ_i^*) on the identity ID_i^*

(2) $\mathcal{A}_{\text{I}}$ has never made an Extract-Partial-Private-Key query on identity ID_i^*

(3) $\mathcal{A}_{\text{I}}$ has never made a Sign query with input $(\text{I D}_{\text{i}}^*, m_i^*)$

3.3.2. Game II. This game is executed between a challenger $\mathcal{C}_{\text{II}}$ and a Type-II adversary $\mathcal{A}_{\text{II}}$. Similar to Game I, Game II also proceeds in three phases.

(i) *Setup*. Like Game I, $\mathcal{C}_{\text{II}}$ operates the Setup algorithm to generate s_{kgc} and params. $\mathcal{C}_{\text{II}}$ sends the params and s_{kgc} to $\mathcal{A}_{\text{II}}$

(ii) *Query*. As in Game I, $\mathscr{A}_{\mathrm{II}}$ adaptively make queries to Create-User, Extract-Partial-Private-Key, Extract-Secret-Value, Replace-Public-Key and Sign oracles. $\mathscr{C}_{\mathrm{II}}$ responds to these queries similarly to Game I

(iii) *Forgery*. $\mathscr{A}_{\mathrm{II}}$ outputs a tuple $(m_i^*, \sigma_i^*, \mathrm{ID}_i^*)$. The $\mathscr{A}_{\mathrm{II}}$ wins the game if the following four conditions are satisfied:

(1) $\mathscr{A}_{\mathrm{II}}$ outputed a rightful message-signature pair (m_i^*, σ_i^*) on the identity ID_i^*

(2) $\mathscr{A}_{\mathrm{II}}$ has never submitted an Extract-Secret-Value query on the challenged identity ID_i^*

(3) $\mathscr{A}_{\mathrm{II}}$ has never submitted a Replace-Public-Key query on the challenged identity ID_i^*

(4) $\mathscr{A}_{\mathrm{II}}$ has never made a Sign query with input (ID_i^*, m_i^*)

3.3.3. Game III. This game is executed between a challenger $\mathscr{C}_{\mathrm{III}}$ and a Type-III adversary $\mathscr{A}_{\mathrm{III}}$. Similar to Game I and Game II, Game III also proceeds in three phases.

(i) *Setup*. Like Game I, $\mathscr{C}_{\mathrm{III}}$ operates the Setup algorithm to generate s_{kgc} and params. $\mathscr{C}_{\mathrm{III}}$ sends the params and s_{kgc} to $\mathscr{A}_{\mathrm{III}}$

(ii) *Query*. As in Game I, $\mathscr{A}_{\mathrm{III}}$ adaptively make queries to Create-User, Extract-Partial-Private-Key, Extract-Secret-Value, Replace-Public-Key, and Sign oracles. $\mathscr{C}_{\mathrm{III}}$ responds to these queries similarly to Game I

(iii) *Forgery*. $\mathscr{A}_{\mathrm{III}}$ outputs a tuple $(m_i^*, \sigma_i^*, \mathrm{ID}_i^*, P_{\mathrm{i}}^*)$. The $\mathscr{A}_{\mathrm{III}}$ wins the game if the following four conditions are satisfied:

(1) $\mathscr{A}_{\mathrm{III}}$ outputed a rightful message-signature pair (m_i^*, σ_i^*) on the identity ID_i^*

(2) $\mathscr{A}_{\mathrm{III}}$ has never submitted an Extract-Secret-Value query on the challenged identity ID_i^*

(3) $\mathscr{A}_{III}$ has never made a Sign query with input (ID_i^*, m_i^*)

(4) The user with ID_i^* cannot repudiate σ_i^*

Definition 1. A PF-CLS scheme is said to be EUF-CMA satisfying KGC trust level 3, if for any polynomial-time Type-I/Type-II/Type-III adversary $\mathscr{A}_{\mathrm{I}}/\mathscr{A}_{\mathrm{II}}/\mathscr{A}_{\mathrm{III}}$, their advantage in winning game I/II/III is negligible.

4. Revisiting Xu et al.'s Scheme

We take Xu et al.'s scheme [17] as an example to illustrate that it cannot achieve KGC trust level 3. The scheme is described as follows:

(i) *Setup:* input the security parameter $k \in Z^+$, select q -order additive group G, where P is a generator of G. KGC randomly selects $s_{\mathrm{kgc}} \in Z_q^*$ as the master key, calculates $P_{\mathrm{pub}} = s_{\mathrm{kgc}}P$, and defines three secure Hash functions: $H_i = \{0,1\}^* \longrightarrow Z_q^*, i = 1, 2, 3$. Finally, KGC publishes the system parameters $\mathrm{params} = k, q, P, G_q, P_{\mathrm{pub}}, H_i$ and keeps the master key s_{kgc} in secret

(ii) *Partial Private Key Extract:* KGC generates PPK of a user i with $ID_i \in \{0,1\}^*$, as follows:

(1) Choose a random value $r_i \in Z_q^*$ and compute $R_i = r_iP$

(2) Computes $h_{1i} = H_1(\mathrm{ID}_i, R_i, P_{\mathrm{pub}})$, $d_i = r_i + s_{\mathrm{kgc}} h_{1i} \bmod q$

(3) KGC sends $D_i = (d_i, R_i)$ as PPK to the user through a secure channel

(4) The user can validate the PPK by verifying the equation $d_iP = R_i + h_{1i}P_{\mathrm{pub}}$

(iii) *Set-Secret-Value*: the user selects the secret value $x_i \in Z_q^*$ randomly and keeps it secretly. Also, the user computes $X_i = x_iP$

(iv) *Set-Public/Private-Key*: the user sets his/her public key as $P_i = (R_i, X_i)$ and private key as $s_i = (d_i, x_i)$

(v) *Sign*: input params, signer's identity ID_i, signing key pair (P_i, s_i) and message m_i. The signer generates a signature σ_i on message m_i as follows:

(1) Choose a random $u_i \in Z_q^*$ and compute $U_i = u_iP$

(2) Compute $h_{2i} = H_2(\mathrm{ID}_i, P_i, P_{\mathrm{pub}})$, $h_{3i} = H_3(\mathrm{ID}_i, m_i, P_i, U_i)$, and $v_i = d_i + h_{2i}x_i + h_{3i}u_i \bmod q$

(3) Set $\sigma_i = (U_i, v_i)$ as the signature of message m_i

(vi) *Verify:* on the input of params, ID_i, P_i, signature σ_i and message m_i, any verifier can verify the signature σ_i on m_i as follows:

(1) Compute $h_{1i} = H_1(\mathrm{ID}_i, R_i, P_{\mathrm{pub}})$, $h_{2i} = H_2(\mathrm{ID}_i, P_i, P_{\mathrm{pub}})$, and $h_{3i} = H_3(\mathrm{ID}_i, m_i, P_i, U_i)$

(2) Verify the equation

$$v_iP = R_i + h_{1i}P_{\mathrm{pub}} + h_{2i}X_i + h_{3i}U_i \tag{1}$$

If equation (1) holds, the verifier outputs "Accept"; else it outputs "Reject".

5. Attack on Xu et al.'s Scheme

In Xu et al.'s scheme [17], if KGC leaks the user's PPK $D_i = (d_i, R_i)$ to the adversary $\mathscr{A}$, $\mathscr{A}$ can successfully forge

the signature by replacing the public key. The specific attack description is as below:

(i) *Replace Public Key*: adversary $\mathscr{A}$ completes the public key replacement by performing the following operations:

(1) Randomly choose $x_i^* \in Z_q^*$ as secret value, compute $X_i^* = x_i^* P$

(2) Replace the original public key with $P_i^* = (R_i, X_i^*)$ of the user ID_i

(ii) *Signature forgery*: to forge the signature of the user ID_i on the message $m_i^* \in \{0,1\}^*$, $\mathscr{A}$ performs as follows:

(1) Randomly choose $u_i^* \in Z_q^*$ and compute $U_i^* = u_i^* P$

(2) Compute $h_{2i}^* = H_2(\mathrm{ID}_i, P_i^*, P_{\mathrm{pub}})$, $h_{3i}^* = H_3(\mathrm{ID}_i, m_i^*, P_i^*, U_i^*)$, $v_i^* = d_i + h_{2i}^* x_i^* + h_{3i}^* u_i^* \mod q$

(3) Output the forged signature $\sigma_i^* = (U_i^*, v_i^*)$

(iii) *Verify*: given the identity $\mathrm{ID}_i \in \{0,1\}^*$, $P_i^* = (R_i, X_i^*)$, m_i^* and $\sigma_i^* = (U_i^*, v_i^*)$, a verifier computes as below:

(1) Compute, $h_{1i} = H_1(\mathrm{ID}_i, R_i, P_{\mathrm{pub}})$, $h_{2i}^* = H_2(\mathrm{ID}_i, P_i^*, P_{\mathrm{pub}})$, and $h_{3i}^* = H_3(\mathrm{ID}_i, m_i^*, P_i^*, U_i^*)$

(2) Verify the equation

$$v_i^* P = R_i + h_{1i} P_{\mathrm{pub}} + h_{2i}^* X_i^* + h_{3i}^* U_i^* \tag{2}$$

Obviously, the forged signature is valid because the verification equation (2) always holds. Because we have

$$\begin{aligned} v_i^* P &= (d_i + h_{2i}^* x_i^* + h_{3i}^* u_i^*) P = (r_i + s h_{1i} + h_{2i}^* x_i^* + h_{3i}^* u_i^*) \\ &\cdot P = R_i + h_{1i} P_{pub} + h_{2i}^* X_i^* + h_{3i}^* U_i^*. \end{aligned} \tag{3}$$

From the above process, we find that in the event of such an attack, any adversary in possession of the d_i can forge X_i^*, so KGC can initiate an attack without being implicated. The KGC might even claim that the user has replaced its original X_i with a new X_i^*. Therefore, Xu et al.'s scheme cannot attain KGC trust level 3, and the schemes in [13–16, 18] can also be proved by similar methods.

6. Our Proposed PF-CLS Scheme

To address the shortcomings of Xu et al.'s scheme, we propose an improved PF-CLS, which consists of seven algorithms and is described as follows. To improve readability, we provide a graphical representation of the scheme as shown in Figure 2.

(i) *Setup*: input the security parameter $k \in Z^+$, select q-order additive group G_q, where P is a generator of G_q. KGC selects the value $s_{\mathrm{kgc}} \in Z_q^*$ randomly, calculates $P_{\mathrm{pub}} = s_{\mathrm{kgc}} P$, and defines three secure Hash functions: $H_i = \{0,1\}^* \longrightarrow Z_q^*, i = 1, 2, 3$. Finally, KGC publishes the system parameters $\mathrm{params} = k, q, P, G_q, P_{\mathrm{pub}}, H_i$ and keeps the master secret key s_{kgc} secretly

(ii) *Set-Secret-Value*: the user selects the secret value $x_i \in Z_q^*$ randomly and computes $X_i = x_i P$ first, then sends (X_i, ID_i) to KGC through secure channel and keeps (x_i, X_i) in secret

(iii) *Partial-Private-Key-Extract*: KGC generates PPK of a user i with $\mathrm{ID}_i \in \{0,1\}^*$, as follows:

(1) Choose a random value $r_i \in Z_q^*$, compute $R_i = r_i P$ and $Q_i = R_i + h_{2i} X_i$, where $h_{2i} = H_2(\mathrm{ID}_i, X_i)$

(2) Calculate $h_{1i} = H_1(\mathrm{ID}_i, Q_i, P_{\mathrm{pub}})$ and $d_i = r_i + s_{\mathrm{kgc}} h_{1i} \mod q$

(3) Send (d_i, Q_i) as PPK to the user through secure channel

(4) The user can validate the PPK by verifying the equation $d_i P = Q_i - h_{2i} X_i + h_{1i} P_{\mathrm{pub}}$

(iv) *Set-Private-Key*: the user sets his/her private key as $s_i = d_i + h_{2i} x_i$, where $h_{2i} = H_2(\mathrm{ID}_i, X_i)$

(v) *Set-Public-Key*: the user sets his/her public key as $P_i = Q_i$

(vi) *Sign*: signer with identity ID_i generates a signature on message $m_i \in \{0,1\}^*$, as follows

(1) Choose a random value $u_i \in Z_q^*$ and compute $U_i = u_i P$

(2) Compute $h_{3i} = H_3(\mathrm{ID}_i, m_i, Q_i, U_i, P_{\mathrm{pub}})$ and $v_i = u_i + h_{3i} s_i \mod q$

(3) Set $\sigma_i = (U_i, v_i)$ as the signature of message m_i

(vii) *Verify*: on inputting of params, ID_i, $P_i = Q_i$, signature σ_i, and message m_i, any verifier can verify the signature σ_i on m_i as follows:

(1) Compute $h_{1i} = H_1(\mathrm{ID}_i, Q_i, P_{\mathrm{pub}})$, $h_{3i} = H_3(\mathrm{ID}_i, m_i, Q_i, U_i, P_{\mathrm{pub}})$

(2) Verify the equation

$$v_i P = U_i + h_{3i}\left(Q_i + h_{1i} P_{\mathrm{pub}}\right) \tag{4}$$

If equation (4) holds, the verifier outputs "Accept"; else it outputs "Reject."

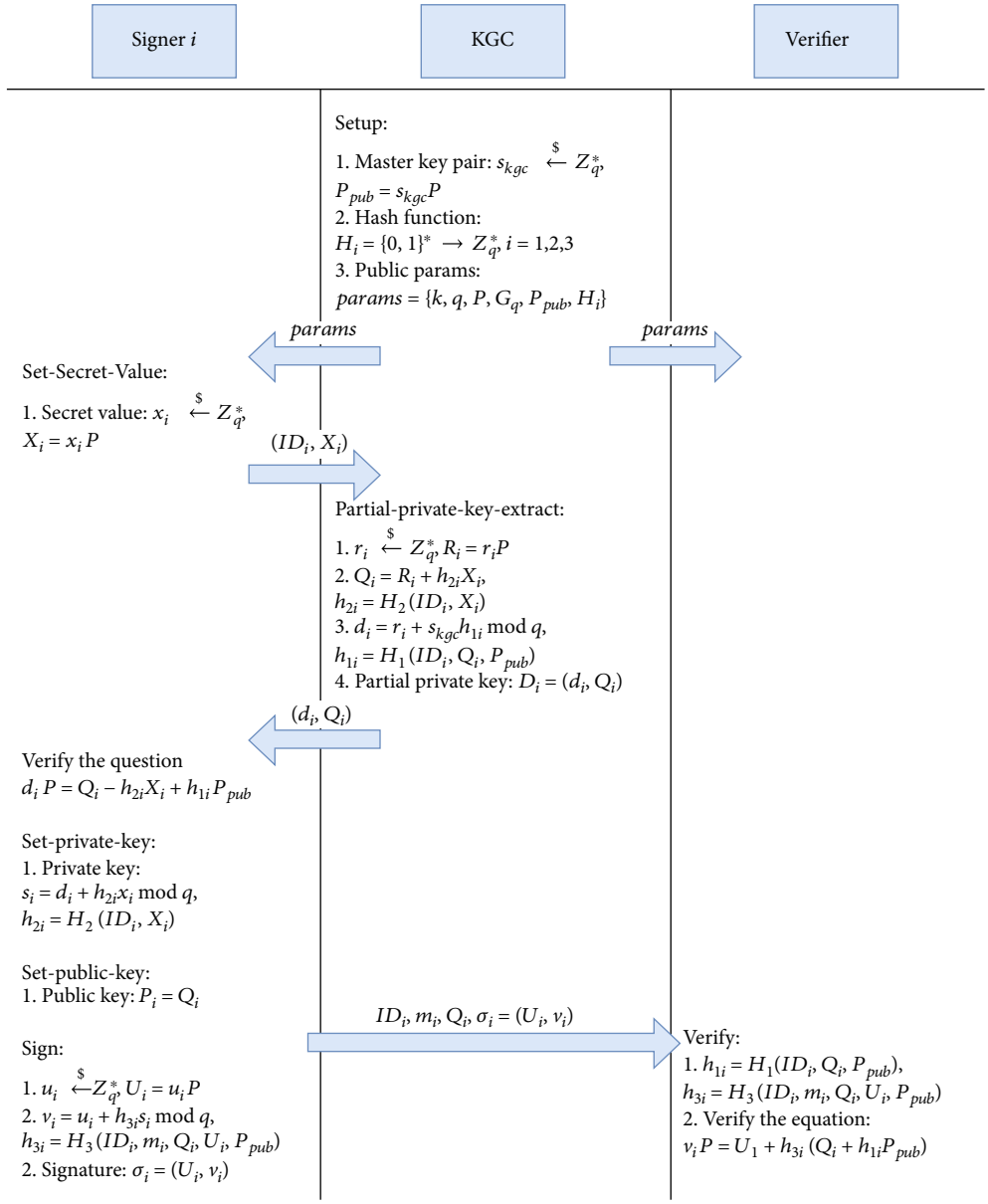

Figure 2: The process of our PF-CLS scheme.

Table 2: Running time of cryptographic operations (ms).

Notations	Operations	Running time
T_{ma}	A modular addition operation	0.000675
T_{mm}	A modular multiplication operation	0.001871
T_{inv}	A modular inversion operation	0.004239
T_{hs}	A general hash operation	0.004010
T_{pa}	A point addition operation	0.001820
T_{pm}	A point multiplication operation	0.316700

As mentioned above, we set $s_i = d_i + h_{2i}x_i$, and the full private key is computed from the PPK d_i and the user's secret value x_i, unlike many PF-CLS schemes where the full private key is denoted as: $s_i = (d_i, x_i)$. Furthermore, the signature's form is $v_i = u_i + h_{3i}s_i$, which corresponds to the Schnorr algorithm. In the following, we briefly give the application of our scheme to aggregate signatures to show the scalability of our scheme.

(i) *Aggregate*: inputting n signatures $((m_1, \sigma_1 = (U_1, v_1)), \cdots, (m_n, \sigma_n = (U_n, v_n)))$ for n users $\mathrm{ID}_1, \cdots, \mathrm{ID}_n$ and computes $v = \sum_{i=1}^{n} v_i$. Then, the algorithm outputs the aggregate signature $\sigma = (U_1, \cdots, U_n, v)$

(ii) *Aggregate verify*: on inputting of params, $\{\mathrm{ID}_1, \cdots, \mathrm{ID}_n\}$, $\{P_1, \cdots, P_n\}$ and signature σ. Any verifier can verify the signature σ on the message $\{m_1, \cdots, m_n\}$ as follows:

(1) Compute $h_{1i} = H_1(\mathrm{ID}_i, Q_i, P_{\mathrm{pub}})$, $h_{3i} = H_3(\mathrm{ID}_i, m_i, Q_i, U_i, P_{\mathrm{pub}})$, where $1 \le i \le n$

(2) Verify the equation $vP = \sum_{i=1}^{n} U_i + \sum_{i=1}^{n} h_{3i}(Q_i + h_{1i}P_{\mathrm{pub}})$, if the equation holds, the verifier outputs "Accept"; else it outputs "Reject."

7. Analysis of our CLS Scheme

7.1. Correctness Proof. Suppose $\sigma_i = (U_i, v_i)$ is the signature produced by our proposed PF-CLS scheme, it is easy to verify that equation (4) holds. The correctness of the proposed scheme can be justified by verifying the equation as follows:

$$\begin{aligned} v_i P &= (u_i + h_{3i}(d_i + h_{2i}x_i))P = U_i + h_{3i}\left(R_i + h_{2i}X_i + h_{1i}P_{\mathrm{pub}}\right) \\ &= U_i + h_{3i}\left(Q_i + h_{1i}P_{\mathrm{pub}}\right). \end{aligned} \tag{5}$$

7.2. Security Analysis. In this section, we demonstrate that our presented PF-CLS scheme is existential unforgeable against the Type-I/II/III adversaries. The security proof of our scheme is described as follows.

Theorem 2. *In the random oracle model, suppose $\mathscr{A}_I$ is a Type-I adversary of probabilistic polynomial time. If $\mathscr{A}_I$ has a nonnegligible advantage ε to forge a rightful signature in Game I after querying at most q_{H_1} times Hash oracle H_1 and q_{ppk} times Extract-Partial-Private-Key oracle, there exists an algorithm $\mathscr{C}_I$ which can call $\mathscr{A}_I$ as a subprogram to figure out the solution to ECDLP with a probability $\varepsilon' \ge (1 - (1/q_{H_1}))^{q_{ppk}} \cdot (1/q_{H_1}) \cdot \varepsilon$.*

Proof. Assume $(P, \alpha P)$ is an arbitrary instance of ECDLP. The purpose of $\mathscr{C}_I$ is to get the solution $\alpha \in Z_q^*$ to the ECDLP by making interaction with $\mathscr{A}_{\mathrm{I}}$

(i) *Setup*. $\mathscr{C}_{\mathrm{I}}$ sets $P_{\mathrm{pub}} = \alpha$P, produces the system public parameters $\mathrm{params} = p, q, P, G_q, P_{\mathrm{pub}}, H_i, i = 1, 2, 3$. Then, $\mathscr{C}_I$ randomly selects an identity $\mathrm{ID}_i^* \in \{0,1\}^*$ as the challenged identity and sends params to $\mathscr{A}_{\mathrm{I}}$. $\mathscr{C}_{\mathrm{I}}$ keeps three lists L_{Cuser}, L_2, and L_3, which are utilized to write down Create-User queries, H_2 queries and H_3 queries, respectively. All lists are initially empty

(ii) *Query*

(a) *Create-User*: when this request is issued on an identity ID_i, $\mathscr{C}_I$ calculates the following:

(1) If $\mathrm{ID}_i \ne \mathrm{ID}_i^*$, $\mathscr{C}_I$ selects random elements $d_i, x_i, h_{1i}, h_{2i} \in Z_q^*$, computes $X_i = x_i P$, $Q_i = d_i P + h_{2i}X_i - h_{1i}P_{\mathrm{pub}}$ and sets $h_{1i} = H_1(\mathrm{ID}_i, Q_i, P_{\mathrm{pub}})$, $h_{2i} = H_2(\mathrm{ID}_i, X_i)$

(2) If $\mathrm{ID}_i = \mathrm{ID}_i^*$, $\mathscr{C}_I$ selects random elements $r_i, x_i, h_{1i}, h_{2i} \in Z_q^*$, computes $X_i = x_i P$, $R_i = r_i P$, $Q_i = R_i + h_{2i}X_i$, and then sets $h_{1i} = H_1(\mathrm{ID}_i, Q_i, P_{\mathrm{pub}})$, $h_{2i} = H_2(\mathrm{ID}_i, X_i)$, $d_i = \perp$

In both cases, the user ID_i has been created. The user's PPK is $D_i = (Q_i, d_i)$, and his/her secret value is x_i. Next, $\mathscr{C}_{\mathrm{I}}$ outputs $P_i = Q_i$ as the public key of the user ID_i to $\mathscr{A}_{\mathrm{I}}$. Then, $\mathscr{C}_{\mathrm{I}}$ adds the tuple $(\mathrm{ID}_i, d_i, x_i, P_i = Q_i, P_{\mathrm{pub}}, h_{1i})$ to the list L_{Cuser} and the tuple $(\mathrm{ID}_i, X_i, h_{2i})$ to the list L_2

(Note: we suppose that the Create-User query always precedes other oracle queries.)

(b) *Extract-Partial-Private-Key*: when $\mathscr{A}_I$ issues this query on a created user ID_i, if $\mathrm{ID}_i \ne \mathrm{ID}_i^*$, $\mathscr{C}_{\mathrm{I}}$ finds the relevant record $(\mathrm{ID}_i, d_i, x_i, Q_i, P_{\mathrm{pub}}, h_{1i})$ from L_{Cuser}, and returns d_i to $\mathscr{A}_I$. Otherwise, $\mathscr{C}_{\mathrm{I}}$ aborts

(c) *Extract-Secret-Value*: when $\mathscr{A}_I$ issues this query on a created user ID_i, $\mathscr{C}_{\mathrm{I}}$ finds the relevant record $(\mathrm{ID}_i, d_i, x_i, Q_i, P_{\mathrm{pub}}, h_{1i})$ from L_{Cuser} and returns x_i to $\mathscr{A}_{\mathrm{I}}$

(d) *Replace-Public-Key*: when receiving a query on input (ID_i, Q_i^*), $\mathscr{C}_I$ retrieves the tuple $(\mathrm{ID}_i, d_i, x_i, Q_i, P_{\mathrm{pub}}, h_{1i})$ from L_{Cuser} and sets $Q_i = Q_i^*$, $x_i = \perp$, $d_i = \perp$, and then renews the abovementioned item $(\mathrm{ID}_i, \perp, \perp, Q_i^*, P_{\mathrm{pub}}, h_{1i})$ to L_{Cuser}

(e) H_1 *Queries*: $\mathscr{A}_{\mathrm{I}}$ submits a tuple $(\mathrm{ID}_i, Q_i, P_{\mathrm{pub}})$ to this oracle. $\mathscr{C}_{\mathrm{I}}$ recovers the corresponding record $(\mathrm{ID}_i, d_i, x_i, Q_i, P_{\mathrm{pub}}, h_{1i})$ from L_{Cuser} and returns h_{1i} to $\mathscr{A}_{\mathrm{I}}$ if it exists. Otherwise, $\mathscr{C}_I$ asks Create-User query and extracts h_{1i} from L_{Cuser} and returns it to $\mathscr{A}_I$

(f) H_2 *Queries*: $\mathscr{C}_{\mathrm{I}}$ keeps a list $L_2 : (\mathrm{ID}_i, X_i, h_{2i})$. When $\mathscr{A}_{\mathrm{I}}$ submits a tuple (ID_i, X_i) to this oracle, $\mathscr{C}_I$ recovers the corresponding record $(\mathrm{ID}_i, X_i, h_{2i})$ from L_2 and returns h_{2i} to $\mathscr{A}_{\mathrm{I}}$ if it exists. Otherwise, $\mathscr{C}_I$ asks Create-User query and returns h_{2i} to $\mathscr{A}_{\mathrm{I}}$

(g) H_3 *Queries*: $\mathscr{C}_{\mathrm{I}}$ keeps a list $L_3 : (\mathrm{ID}_i, m_i, Q_i, U_i, P_{\mathrm{pub}}, h_{3i})$. On inputting an item $(\mathrm{ID}_i, m_i, Q_i, U_i, P_{\mathrm{pub}})$, $\mathscr{C}_{\mathrm{I}}$ finds the list L_3. If it contains the relevant tuple $(\mathrm{ID}_i, m_i, Q_i, U_i, P_{\mathrm{pub}}, h_{3i})$, $\mathscr{C}_{\mathrm{I}}$ outputs h_{3i} to $\mathscr{A}_{\mathrm{I}}$. Otherwise, $\mathscr{C}_{\mathrm{I}}$ randomly selects an element $h_{3i} \in Z_q^*$ and sets $h_{3i} = H_3(\mathrm{ID}_i, m_i, Q_i, U_i, P_{\mathrm{pub}})$ to $\mathscr{A}_{\mathrm{I}}$, and inserts $(\mathrm{ID}_i, m_i, Q_i, U_i, P_{\mathrm{pub}}, h_{3i})$ to L_3

(h) *Sign*: when $\mathscr{C}_{\mathrm{I}}$ receiving this query on inputs a message m_i, an identity ID_i, and the current public key $P_i = Q_i$, $\mathscr{C}_{\mathrm{I}}$ recovers $L_{\mathrm{Cuser}} : (\mathrm{ID}_i, d_i, x_i, Q_i, P_{\mathrm{pub}}, h_{1i})$, $L_2 : (\mathrm{ID}_i, X_i, h_{2i})$ and calculates the following:

(1) If $\mathrm{ID}_i \ne \mathrm{ID}_i^*$ and P_i has not been replaced, $\mathscr{C}_{\mathrm{I}}$ randomly selects $u_i, h_{3i} \in Z_q^*$ and computes $U_i = u_i P$ and sets $h_{3i} = H_3(\mathrm{ID}_i, m_i, Q_i, U_i, P_{\mathrm{pub}})$ and computes $v_i = u_i + h_{3i}(d_i + h_{2i}x_i) \bmod q$. $\mathscr{C}_I$ sends $\sigma_i = (U_i, v_i)$ as a signature to $\mathscr{A}_I$, and then inserts $(\mathrm{ID}_i, m_i, Q_i, U_i, P_{\mathrm{pub}}, h_{3i})$ into L_3

(2) Otherwise, $\mathscr{C}_{\mathrm{I}}$ randomly selects $v_i, h_{3i} \in Z_q^*$ and computes $U_i = v_i P - h_{3i}(Q_i + h_{1i}P_{\mathrm{pub}})$, and sets $h_{3i} = H_3(\mathrm{ID}_i, m_i, Q_i, U_i, P_{\mathrm{pub}})$. $\mathscr{C}_{\mathrm{I}}$ returns $\sigma_i = (U_i, v_i)$ as a signature to $\mathscr{A}_{\mathrm{I}}$. Then, adds the item $(\mathrm{ID}_i, m_i, Q_i, U_i, P_{\mathrm{pub}}, h_{3i})$ to L_3

(iii) *Forgery*. At last, $\mathscr{A}_{\mathrm{I}}$ outputs a rightful message/signature tuple $(m_i^*, \sigma_i^* = (U_i^*, v_i^*))$ for ID_i^* with $P_i^* = Q_i^*$ which may be replaced by $\mathscr{A}_{\mathrm{I}}$. If $\mathrm{ID}_i \neq \mathrm{ID}_i^*$, $\mathscr{C}_I$ aborts, or else, $\mathscr{C}_{\mathrm{I}}$ recovers the list $L_{\mathrm{Cuser}} : (\mathrm{ID}_i^*, d_i^*, x_i^*, Q_i^*, P_{\mathrm{pub}}, h_{1i}^*)$, $L_2 : (\mathrm{ID}_i^*, X_i^*, h_{2i}^*)$ and $L_3 : (\mathrm{ID}_i^*, m_i^*, Q_i^*, U_i^*, P_{\mathrm{pub}}, h_{3i}^*)$, respectively

We apply the forking lemma [29] in the following simulation. $\mathscr{C}_{\mathrm{I}}$ replays $\mathscr{A}_{\mathrm{I}}$ with the same random tape but provides another different value of H_1. That is, $h_{1i}^{*(2)} = H_1(\mathrm{ID}_i^*, Q_i^*, P_{\mathrm{pub}})$, and $h_{1i}^* \neq h_{1i}^{*(2)}$. Then, $\mathscr{A}_{\mathrm{I}}$ outputs another two valid signature $\sigma_i^{*(2)} = (U_i^*, v_i^{*(2)})$ on the same message m_i^*.

Hence, we have the following two equations (for convenience, we let $h_{1i}^{*(1)} = h_{1i}^*, v_i^{*(1)} = v_i^*$):

$$\begin{aligned} v_i^{*(j)} P &= U_i^* + h_{3i}^* \left(Q_i^{*(j)} + h_{1i}^* P_{\mathrm{pub}} \right), j = 1, 2 \Rightarrow v_i^{*(j)} \\ &= u_i^* + h_{3i}^* \left(r_i + h_{2i} x_i + h_{1i}^{*(j)} \alpha \right), j = 1, 2. \end{aligned} \tag{6}$$

In the above two equations (6), u_i^* and α are unknown for $\mathscr{C}_I$. Hence, $\mathscr{C}_{\mathrm{I}}$ can successfully obtain the value of α by solving the equation system. That is,

$$\alpha = \frac{v_i^{*(1)} - v_i^{*(2)}}{h_{3i}^* h_{1i}^{*(1)} - h_{3i}^* h_{1i}^{*(2)}}. \tag{7}$$

However, this contradicts the ECDLP assumption. Namely, the signature $\sigma_i = (U_i, v_i)$ cannot be forged by $\mathscr{A}_{\mathrm{I}}$.

Next, let us calculate $\mathscr{C}_I$'s winning probability in Game I. When events Δ_1, Δ_2, and Δ_3 occur, $\mathscr{C}_{\mathrm{I}}$ will win this game.

Δ_1: $\mathscr{C}_{\mathrm{I}}$ does not abort the game when $\mathscr{A}_I$ queries Extract-Partial-Private-Key oracle

Δ_2: in the forgery phase, $\mathscr{A}_I$ outputs a message-signature pair (m_i^*, σ_i^*) on an identity $\mathrm{ID}_i = \mathrm{ID}_i^*$

Δ_3: σ_i^* is a valid forgery on (m_i^*, ID_i^*)

Obviously,

$$\begin{aligned} P[\Delta_1] &\geq \left(1 - \frac{1}{q_{H_1}}\right)^{q_{\mathrm{ppk}}}, \\ P[\Delta_2|\Delta_1] &\geq \frac{1}{q_{H_1}}, \\ P[\Delta_3|\Delta_1 \cap \Delta_2] &= \varepsilon. \end{aligned} \tag{8}$$

As a result, $\mathscr{C}_{\mathrm{I}}$'s probability is

$$\begin{aligned} \varepsilon' &= P[\Delta_1 \cap \Delta_2 \cap \Delta_3] = P[\Delta_1] P[\Delta_2|\Delta_1] P[\Delta_3|\Delta_1 \cap \Delta_2] \\ &\geq \left(1 - \frac{1}{q_{H_1}}\right)^{q_{\mathrm{ppk}}} \cdot \frac{1}{q_{H_1}} \cdot \varepsilon. \end{aligned} \tag{9}$$

Hence, $\mathscr{C}_{\mathrm{I}}$ handles the ECDLP with a probability $\varepsilon' \geq (1 - (1/q_{H_1}))^{q_{ppk}} \cdot (1/q_{H_1}) \cdot \varepsilon$. □

Theorem 3. *In the random oracle model, suppose $\mathscr{A}_{II}$ is a Type-II adversary of probabilistic polynomial time. If $\mathscr{A}_{II}$ has a nonnegligible advantage ε to forge a rightful signature in Game II after querying at most q_{H_1} times Hash oracle H_1 and q_{esv} times Extract-Secret-Value oracle and q_{rpk} times Replace-Public-Key oracle, there exists an algorithm $\mathscr{C}_{II}$ which can call $\mathscr{A}_{II}$ as a subprogram to figure out the solution to ECDLP with a probability $\varepsilon' \geq (1 - (1/q_{H_1}))^{q_{esv}+q_{rpk}} \cdot (1/q_{H_1}) \cdot \varepsilon$.*

Proof. Assume $(P, \alpha P)$ is an arbitrary instance of ECDLP. The purpose of $\mathscr{C}_{\mathrm{II}}$ is to get the solution $\alpha \in Z_q^*$ to the ECDLP by making interaction with $\mathscr{A}_{\mathrm{II}}$:

(i) *Setup*. $\mathscr{C}_{\mathrm{II}}$ randomly selects $s_{\mathrm{kgc}} \in Z_q^*$ and calculates $P_{\mathrm{pub}} = s_{\mathrm{kgc}} P$, produces the system public parameters $\mathrm{params} = p, q, P, G_q, P_{\mathrm{pub}}, H_i, i = 1, 2, 3$. Then, $\mathscr{C}_{\mathrm{II}}$ randomly selects an identity $\mathrm{ID}_i^* \in \{0, 1\}^*$ as the challenged identity and sends $(\mathrm{params}, s_{\mathrm{kgc}})$ to $\mathscr{A}_{\mathrm{II}}$. $\mathscr{C}_{\mathrm{II}}$ keeps three lists L_{Cuser}, L_2, and L_3, which are utilized to write down Create-User queries, H_2 queries and H_3 queries, respectively. All lists are initially empty

(ii) *Query*

(a) *Create-User*: when this request is issued on an identity ID_i, $\mathscr{C}_{\mathrm{II}}$ calculates the following:

(1) If $\mathrm{ID}_i \neq \mathrm{ID}_i^*$, $\mathscr{C}_{\mathrm{II}}$ selects random elements r_i, $x_i, h_{1i}, h_{2i} \in Z_q^*$, computes $X_i = x_i P, R_i = r_i P$, $Q_i = R_i + h_{2i} X_i$, and sets $h_{1i} = H_1(\mathrm{ID}_i, Q_i, P_{\mathrm{pub}})$, $h_{2i} = H_2(\mathrm{ID}_i, X_i)$, and then computes $d_i = r_i + s h_{1i} \bmod q$

(2) If $\mathrm{ID}_i = \mathrm{ID}_i^*$, $\mathscr{C}_{\mathrm{II}}$ selects random elements r_i, $h_{1i}, h_{2i} \in Z_q^*$, sets $X_i = \alpha P$, and computes $d_i = r_i + s h_{1i} \bmod q$, $R_i = r_i P$, $Q_i = R_i + h_{2i} X_i$, and then sets $h_{1i} = H_1(\mathrm{ID}_i, Q_i, P_{\mathrm{pub}})$, $h_{2i} = H_2(\mathrm{ID}_i, X_i)$, $x_i = \perp$

In both cases, the user ID_i has been created. The user's PPK is $D_i = (Q_i, d_i)$, and his/her secret value is x_i. Next, $\mathscr{C}_{\mathrm{II}}$ outputs $P_i = Q_i$ as the public key of the user ID_i to $\mathscr{A}_{\mathrm{II}}$. Then, $\mathscr{C}_{\mathrm{II}}$ adds the tuple $(\mathrm{ID}_i, d_i, x_i, P_i = Q_i, P_{\mathrm{pub}}, h_{1i})$ to the list L_{Cuser} and the tuple $(\mathrm{ID}_i, X_i, h_{2i})$ to the list L_2.

(Note. We suppose that the Create-User query always precedes other oracle queries)

(b) *Extract-Partial-Private-Key*: when $\mathscr{A}_{\mathrm{II}}$ issues this query on a created user ID_i, $\mathscr{C}_{\mathrm{II}}$ finds the relevant record $(\mathrm{ID}_i, d_i, x_i, Q_i, P_{\mathrm{pub}}, h_{1i})$ from L_{Cuser} and returns d_i to $\mathscr{A}_{\mathrm{II}}$

(c) *Extract-Secret-Value*: when $\mathscr{A}_{II}$ issues this query on a created user ID_i. If $\mathrm{ID}_i \neq \mathrm{ID}_i^*$, $\mathscr{C}_{\mathrm{II}}$ finds the relevant record $(\mathrm{ID}_i, d_i, x_i, Q_i, P_{\mathrm{pub}}, h_{1i})$ from L_{Cuser} and returns x_i to $\mathscr{A}_{\mathrm{II}}$. Otherwise, $\mathscr{C}_{\mathrm{II}}$ aborts

(d) *Replace-Public-Key*: when receiving a query on input (ID_i, Q_i^*), if $\mathrm{ID}_i \neq \mathrm{ID}_i^*$, $\mathscr{C}_{\mathrm{II}}$ retrieves the tuple $(\mathrm{ID}_i, d_i, x_i, Q_i, P_{\mathrm{pub}}, h_{1i})$ from L_{Cuser} and sets $Q_i = Q_i^*$, $x_i = \perp$, $d_i = \perp$, and then renews the abovementioned item $(ID_i, \perp, \perp, Q_i^*, P_{\mathrm{pub}}, h_{1i})$ to L_{Cuser}. Otherwise, $\mathscr{C}_{\mathrm{II}}$ aborts

(e) H_1, H_2, H_3 *Queries*: the answers to H_1, H_2, and H_3 queries are similar to do in Theorem 2

(f) *Sign*: when $\mathscr{C}_{\mathrm{II}}$ receiving this query on inputs a message m_i, an identity ID_i, and the current public key $P_i = Q_i$, $\mathscr{C}_{\mathrm{II}}$ recovers $L_{\mathrm{Cuser}} : (\mathrm{ID}_i, d_i, x_i, Q_i, P_{\mathrm{pub}}, h_{1i})$, $L_2 : (\mathrm{ID}_i, X_i, h_{2i})$ and calculates the following:

(1) If $\mathrm{ID}_i \neq \mathrm{ID}_i^*$ and P_i has not been replaced, $\mathscr{C}_{\mathrm{II}}$ randomly selects $u_i, h_{3i} \in Z_q^*$ and computes $U_i = u_i P$ and sets $h_{3i} = H_3(\mathrm{ID}_i, m_i, Q_i, U_i, P_{\mathrm{pub}})$ and computes $v_i = u_i + h_{3i}(d_i + h_{2i}x_i) \bmod q$. $\mathscr{C}_{\mathrm{II}}$ sends $\sigma_i = (U_i, v_i)$ as a signature to $\mathscr{A}_{\mathrm{II}}$, and then inserts $(\mathrm{ID}_i, m_i, Q_i, U_i, P_{\mathrm{pub}}, h_{3i})$ into L_3

(2) Otherwise, $\mathscr{C}_{\mathrm{II}}$ randomly selects $v_i, h_{3i} \in Z_q^*$ and computes $U_i = v_i P - h_{3i}(Q_i + h_{1i}P_{\mathrm{pub}})$, and sets $h_{3i} = H_3(\mathrm{ID}_i, m_i, Q_i, U_i, P_{\mathrm{pub}})$. $\mathscr{C}_{\mathrm{II}}$ returns $\sigma_i = (U_i, v_i)$ as a signature to $\mathscr{A}_{\mathrm{II}}$. Then, adds the item $(\mathrm{ID}_i, m_i, Q_i, U_i, P_{\mathrm{pub}}, h_{3i})$ to L_3

(iii) *Forgery*. At last, $\mathscr{A}_{\mathrm{II}}$ outputs a rightful message/signature tuple $(m_i^*, \sigma_i^* = (U_i^*, v_i^*))$ for ID_i^* with $P_i^* = Q_i^*$. If $\mathrm{ID}_i \neq \mathrm{ID}_i^*$, $\mathscr{C}_{\mathrm{II}}$ aborts, or else, $\mathscr{C}_{\mathrm{II}}$ recovers the list $L_{\mathrm{Cuser}} : (\mathrm{ID}_i^*, d_i^*, x_i^*, Q_i^*, P_{\mathrm{pub}}, h_{1i}^*)$, $L_2 : (\mathrm{ID}_i^*, X_i^*, h_{2i}^*)$, and $L_3 : (\mathrm{ID}_i^*, m_i^*, Q_i^*, U_i^*, P_{\mathrm{pub}}, h_{3i}^*)$, respectively

We apply the forking lemma [29] in the following simulation. $\mathscr{C}_{\mathrm{II}}$ replays $\mathscr{A}_{\mathrm{II}}$ with the same random tape but provides another different value of H_2. That is, $h_{2i}^{*(2)} = H_2(\mathrm{ID}_i^*, X^*)$, and $h_{2i}^* \neq h_{2i}^{*(2)}$. Then, $\mathscr{A}_{\mathrm{II}}$ outputs another valid signature $\sigma_i^{*(2)} = (U_i^*, v_i^{*(2)})$ on the same message m_i^*

Hence, we have the following two equations (for convenience, we let $h_{2i}^{*(1)} = h_{2i}^*, v_i^{*(1)} = v_i^*$).

$$\begin{aligned} v_i^{*(j)} P &= U_i^* + h_{3i}^* \left(Q_i^{*(j)} + h_{1i}^* P_{pub} \right), j = 1, 2 \Rightarrow v_i^{*(j)} \\ &= u_i^* + h_{3i}^* \left(d_i + h_{2i}^{*(j)} \alpha \right), j = 1, 2. \end{aligned} \tag{10}$$

In the above two equations (10), u_i^* and α are unknown for $\mathscr{C}_{\mathrm{II}}$. Hence, $\mathscr{C}_{\mathrm{II}}$ can successfully obtain the value of α by solving the equation system. That is,

(i)

$$\alpha = \frac{v_i^{*(1)} - v_i^{*(2)}}{h_{3i}^* h_{2i}^{*(1)} - h_{3i}^* h_{2i}^{*(2)}} \tag{11}$$

However, this contradicts the ECDLP assumption. Namely, the signature $\sigma_i = (U_i, v_i)$ cannot be forged by $\mathscr{A}_{\mathrm{II}}$

Next, let us calculate $\mathscr{C}_{\mathrm{II}}$'s winning probability in Game II. When events Δ_1, Δ_2, and Δ_3 occur, $\mathscr{C}_{\mathrm{II}}$ will win this game.

Δ_1: $\mathscr{C}_{\mathrm{II}}$ does not abort the game when $\mathscr{A}_{II}$ queries Extract-Secret-Value and Replace-Public-Key oracle

Δ_2: In the forgery phase, $\mathscr{A}_{II}$ outputs a message-signature pair (m_i^*, σ_i^*) on an identity $ID_i = ID_i^*$

Δ_3: σ_i^* is a valid forgery on (m_i^*, ID_i^*)

Obviously,

$$\begin{aligned} P[\Delta_1] &\geq \left(1 - \frac{1}{q_{H_1}}\right)^{q_{\mathrm{esv}}} \left(1 - \frac{1}{q_{H_1}}\right)^{q_{\mathrm{rpk}}}, \\ P[\Delta_2 | \Delta_1] &\geq \frac{1}{q_{H_1}}, \\ P[\Delta_3 | \Delta_1 \cap \Delta_2] &= \varepsilon. \end{aligned} \tag{12}$$

As a result, $\mathscr{C}_{\mathrm{II}}$'s probability is

$$\begin{aligned} \varepsilon' &= P[\Delta_1 \cap \Delta_2 \cap \Delta_3] = P[\Delta_1] P[\Delta_2 | \Delta_1] P[\Delta_3 | \Delta_1 \cap \Delta_2] \\ &\geq \left(1 - \frac{1}{q_{H_1}}\right)^{q_{\mathrm{esv}} + q_{\mathrm{rpk}}} \cdot \frac{1}{q_{H_1}} \cdot \varepsilon. \end{aligned} \tag{13}$$

Hence, $\mathscr{C}_{\mathrm{II}}$ handles the ECDLP with a probability $\varepsilon' \geq (1 - (1/q_{H_1}))^{q_{\mathrm{esv}} + q_{\mathrm{rpk}}} \cdot (1/q_{H_1}) \cdot \varepsilon$. □

Theorem 4. *In the random oracle model, suppose $\mathscr{A}_{III}$ is a Type-III adversary of probabilistic polynomial time, $\mathscr{A}_{III}$'s advantage in winning Game III is negligible.*

Proof. The challenger $\mathscr{C}_{\mathrm{III}}$ executes Game III with $\mathscr{A}_{\mathrm{III}}$ as follows:

(i) *Setup*. The description is similar to that of Theorem 3, but $\mathscr{C}_{\mathrm{II}}$ is replaced with $\mathscr{C}_{\mathrm{III}}$ and $\mathscr{A}_{\mathrm{II}}$ is replaced with $\mathscr{A}_{\mathrm{III}}$ in the description

(ii) *Query*

(a) *Create-User, Extract-Partial-Private-Key, Extract-Secret-Value, Sign, H_1, H_2, H_3 Queries*: the answers to these queries are similar to do in Theorem 3

(b) *Replace-Public-Key*: when receiving a query on input (ID_i, Q_i^*), $\mathscr{C}_{\mathrm{III}}$ retrieves the tuple $(\mathrm{ID}_i,$

Table 3: A comparative summary: performance (ms).

Scheme	Sign phase	Verify phase	In total
Jia et al. [14]	$1T_{pm} + 2T_{hs} + 1T_{inv} + 2T_{mm} + 2T_{ma}$	$4T_{pm} + 2T_{pa} + 2T_{hs}$	1.612511
Du et al. [15]	$1T_{pm} + 2T_{hs} + 1T_{inv} + 2T_{mm} + 1T_{ma}$	$4T_{pm} + 3T_{pa} + 2T_{hs}$	1.617666
Thumbur et al. [16]	$1T_{pm} + 1T_{hs} + 1T_{mm} + 1T_{ma}$	$3T_{pm} + 2T_{pa} + 2T_{hs}$	1.285016
Xu et al. [17]	$1T_{pm} + 2T_{hs} + 2T_{mm} + 2T_{ma}$	$4T_{pm} + 3T_{pa} + 3T_{hs}$	1.614102
Our scheme	$1T_{pm} + 1T_{hs} + 1T_{mm} + 1T_{ma}$	$3T_{pm} + 2T_{pa} + 2T_{hs}$	1.285016

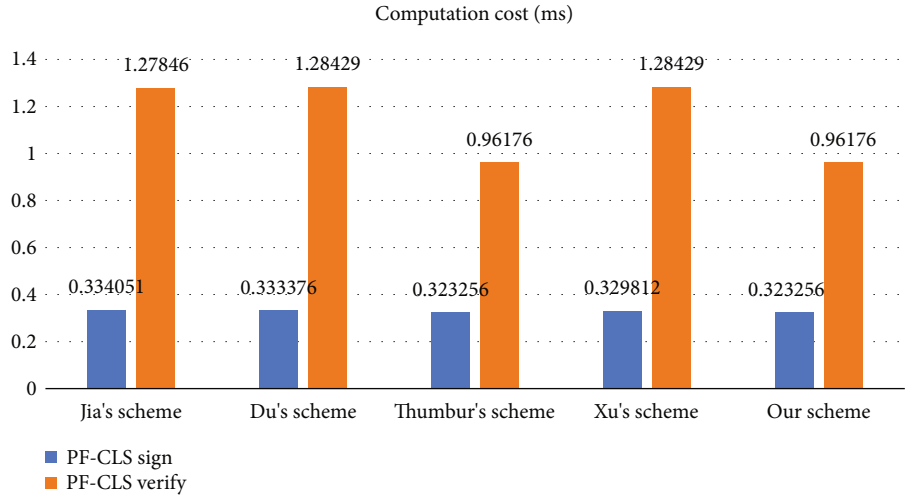

Figure 3: A comparison of computation costs.

Table 4: A comparative summary: security.

Scheme	Security level 2	Security level 3
Jia et al. [14]	No [15]	No
Du et al. [15]	Yes	No
Thumbur et al. [16]	No [17]	No
Xu et al. [17]	Yes	No
Our scheme	Yes	Yes

$d_i, x_i, Q_i, P_{pub}, h_{1i})$ from L_{Cuser} and sets $Q_i = Q_i^*$, $x_i = \perp$, $d_i = \perp$, and then renews the abovementioned item $(ID_i, \perp, \perp, Q_i^*, P_{pub}, h_{1i})$ to L_{Cuser}

(iii) *Forgery*. $\mathcal{A}_{III}$ replaces the public key of the user whose identity is ID_i^*, $P_i = Q_i$ with the corresponding secret value x_i by a new public key $P_i^* = Q_i^*$ with the corresponding secret value x_i^*. Since $\mathcal{A}_{III}$ knows s_{kgc}, it can compute d_i^* corresponding to P_i^*. By the use of x_i^* and d_i^*, $\mathcal{A}_{III}$ can output a rightful signature tuple $(m_i^*, \sigma_i^*, ID_i^*, P_i^*)$

Next, we show that the user is able to repudiate the signature σ_i^*. The user can provide a valid signature σ_i for message m that can be verified by the user's original public key P_i. As a result, the verifier has received two valid signatures σ_i^* and σ_i corresponding to different public keys P_i^* and P_i, respectively. The user denies that the signature σ_i^* was produced by him/her by the following reasons:

(i) According to Theorem 2, σ_i^* cannot be produced by any user other than KGC and the user whose identity is ID_i^*

(ii) The user is able to provide a valid signature σ_i, indicating that the user has the public key P_i and the corresponding PPK d_i. But the public key corresponding to the signature σ_i^* is P_i^*

(iii) In our proposed scheme, the PPK generation method was: $d_i = r_i + s_{kgc}h_{1i}$, $R_i = r_iP$, $X_i = x_iP$, $h_{1i} = H_1(ID_i, P_i = Q_i = R_i + h_{2i}X_i, P_{pub})$, $h_{2i} = H_2(ID_i, X_i)$. If we consider s_{kgc} as the private key and P_{pub} as the public key, this is a schnorr signature form with message (ID_i, X_i, P_{pub}) and signature value (d_i, Q_i). According to the security of the Schnorr signature [28], it is clear that P_i is in one-to-one correspondence with d_i, and since the user does not know s_{kgc}, he/she cannot get d_i^* corresponding to P_i^*

Hence, $\mathcal{A}_{III}$'s advantage in winning Game III is negligible. □

Table 5: A comparison of the communication cost.

Scheme	Jia et al. [14]	Du et al. [15]	Thumbur et al. [16]	Xu et al. [17]	Our scheme
Public key size	$\|G\| + \|G\|$	$\|G\| + \|G\|$	$\|G\| + \|G\|$	$\|G\| + \|G\|$	$\|G\|$
Signature size	$\|G\| + \|Z_q^*\|$	$\|G\| + \|Z_q^*\|$	$\|G\| + \|Z_q^*\|$	$\|G\| + \|Z_q^*\|$	$\|G\| + \|Z_q^*\|$

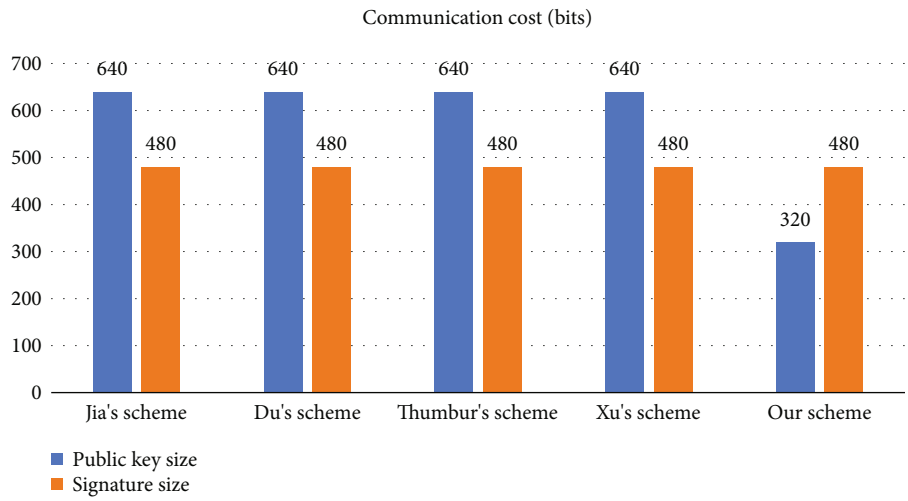

Figure 4: A comparison of communication costs.

8. Performance Evaluation

In this section, we evaluate our PF-CLS scheme from three aspects: computational efficiency, security level, and communication overhead. For this, we choose a nonsingular elliptic curve $E: y^2 = x^3 + \beta x + \lambda \bmod q$, where β, λ are 160-bit primes and run a simulation experiment using the MIRACL library on a personal computer (Intel(R) Core (TM) i5-9300HF CPU @ 2.40GHz, 16.0 GB RAM, and Windows 10 operating system). In the comparison of computational efficiency, the running times of cryptographic operations are shown in Table 2.

8.1. Computation Costs. Due to the characteristics of WSN devices, such as limited computing and processing power, the computational overhead of generating signatures for WSN devices should be as small as possible. In the efficiency analysis of PF-CLS schemes, the computation costs mainly depend on the computation amount of the signature algorithm and verification algorithm. As can be seen from Table 3 and Figure 3, our scheme and Thumbur et al.'s scheme [16] compared with other PF-CLS schemes in [14, 15, 17], the computational efficiency in signature and verification has obvious advantages. However, Table 4 shows that the PF-CLS scheme in [16] cannot resist the attacks of the Type-I adversaries but our scheme falls into KGC trust level 3. That is to say, our new scheme has better computational efficiency and higher security. To sum up, according to the results of all the above experimental analysis and the theoretical analysis in Tables 3 and 4. We conclude that our PF-CLS scheme is more secure and more efficient.

8.2. Communication Costs. Since WSN devices possess limited battery power and communication bandwidth, one of the goals of our PF-CLS scheme is to reduce the communication overhead of WSN devices. The communication cost depends mainly on the size of signature and public key. From Table 5 and Figure 4, the signature size of our scheme is equivalent to that in [14]–[17], which is $|G| + |Z_q^*|$ ($320 + 160 = 480$ bits), where $|G|$ denotes the size of the point in the group and $|Z_q^*|$ denotes the size of the number in Z_q^*. Also, our scheme has a shorter public key size which is $|G|$(320 bits) compared with other schemes [14–17] which is $2|G|$(640 bits). Hence, the proposed CLS scheme has a lower communication overhead.

9. Conclusions

Digital signature technology can provide identity authentication, ensure data integrity, and nonrepudiation. Most WSN devices have limited computing, storage and communication capabilities and require a "lightweight" digital signature scheme to protect data integrity and data authenticity. The PF-CLS scheme requires low computing and storage resources as well as communication bandwidth, making it a suitable choice for WSN devices. However, we found that once the PF-CLS scheme fails to achieve the trust level 3 defined by Girault, the malicious KGC can create false guarantees to impersonate any user in the system without being implicated, which affects the adoption and promotion of the PF-CLS scheme.

In this paper, we took Xu et al.'s scheme as an example and proved that it cannot support the KGC trust level 3. We presented a new PF-CLS scheme with KGC trust level 3. The KGC cannot compute the user's secret keys or generate false guarantees without being implicated. To facilitate the scheme promotion, our signature conforms to the Schnorr signature form. The security analysis presented that our proposed scheme is existentially unforgeable against adaptive chosen-message and identity attacks. The efficiency analysis showed that our PF-CLS scheme with stronger security, lower computational cost, and shorter public key size can be rapidly deployed on hardware and software. At the same time, it has broad application prospects in resource-constrained environments such as the WSN and IoT.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was funded by the Higher Education Department of the Ministry of Education Industry-University Cooperative Education Project grant numbers 201802007011 and 201902169008.

References

[1] D. He, N. Kumar, and J.-H. Lee, "Privacy-preserving data aggregation scheme against internal attackers in smart grids," *Wireless Networks*, vol. 22, no. 2, pp. 491–502, 2016.

[2] N. Kumar, K. Kaur, S. C. Misra, and R. Iqbal, "An intelligent RFID-enabled authentication scheme for healthcare applications in vehicular mobile cloud," *Peer-to-Peer Networking and Applications*, vol. 9, no. 5, pp. 824–840, 2016.

[3] S. Roy, A. K. Das, S. Chatterjee, N. Kumar, S. Chattopadhyay, and J. J. P. C. Rodrigues, "Provably secure fine-grained data access control over multiple cloud servers in mobile cloud computing based healthcare applications," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 1, pp. 457–468, 2019.

[4] N. Temene, C. Sergiou, C. Georgiou, and V. Vassiliou, "A survey on mobility in wireless sensor networks," *Ad Hoc Networks*, vol. 125, article 102726, 2022.

[5] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Advances in Cryptology - ASIACRYPT 2003*, vol. 2894, pp. 452–473, Springer, 2003.

[6] D. H. Yum and P. J. Lee, "Generic construction of certificateless signature," in *Information Security and Privacy*, pp. 200–211, Springer, 2004.

[7] X. Huang, W. Susilo, Y. Mu, and F. Zhang, "On the security of certificateless signature schemes from Asiacrypt 2003," in *Cryptology and Network Security*, pp. 13–25, Springer, 2005.

[8] Q. Mei, Y. Zhao, and H. Xiong, "A new provably secure certificateless signature with revocation in the standard model," *Informatica*, vol. 30, no. 4, pp. 711–728, 2019.

[9] X. Yang, X. Pei, G. Chen, T. Li, M. Wang, and C. Wang, "A strongly unforgeable certificateless signature scheme and its application in IoT environments," *Sensors*, vol. 19, no. 12, p. 2692, 2019.

[10] A. A. Addobea, J. Hou, and Q. Li, "MHCOOS: an offline-online certificateless signature scheme for M-health devices," *Security and Communication Networks*, vol. 2020, Article ID 7085623, 12 pages, 2020.

[11] C. Wu, H. Huang, K. Zhou, and C. Xu, "Cryptanalysis and improvement of a new certificateless signature scheme in the standard model," *China Communications*, vol. 18, no. 1, pp. 151–160, 2021.

[12] D. He, J. Chen, and R. Zhang, "An efficient and provably-secure certificateless signature scheme without bilinear pairings," *International Journal of Communication Systems*, vol. 25, no. 11, pp. 1432–1442, 2012.

[13] K.-H. Yeh, C. Su, K.-K. R. Choo, and W. Chiu, "A novel certificateless signature scheme for smart objects in the internet-of-things," *Sensors*, vol. 17, no. 5, p. 1001, 2017.

[14] X. Jia, D. He, Q. Liu, and K.-K. R. Choo, "An efficient provably-secure certificateless signature scheme for Internet-of- Things deployment," *Ad Hoc Networks*, vol. 71, pp. 78–87, 2018.

[15] H. Du, Q. Wen, S. Zhang, and M. Gao, "A new provably secure certificateless signature scheme for internet of things," *Ad Hoc Networks*, vol. 100, article 102074, 2020.

[16] G. Thumbur, G. S. Rao, P. V. Reddy, N. B. Gayathri, and D. V. R. K. Reddy, "Efficient pairing-free certificateless signature scheme for secure communication in resource-constrained devices," *IEEE Communications Letters*, vol. 24, no. 8, pp. 1641–1645, 2020.

[17] Z. Xu, M. Luo, M. K. Khan, K.-K. R. Choo, and D. He, "Analysis and improvement of a certificateless signature scheme for resource-constrained scenarios," *IEEE Communications Letters*, vol. 25, no. 4, pp. 1074–1078, 2021.

[18] D. Xiang, X. Li, J. Gao, and X. Zhang, "A secure and efficient certificateless signature scheme for Internet of Things," *Ad Hoc Networks*, vol. 124, article 102702, 2022.

[19] M. H. Au, J. Chen, J. K. Liu, Y. Mu, D. S. Wong, and G. Yang, "Malicious KGC attacks in certificateless cryptography," February 14, 2022, https://eprint.iacr.org/2006/255.

[20] X. Huang, Y. Mu, W. Susilo, D. S. Wong, and W. Wu, "Certificateless signature revisited," in *Information Security and Privacy*, pp. 308–322, Springer, Berlin, Heidelberg, 2007.

[21] M. Girault, "Self-certified public keys," in *Advances in Cryptology — EUROCRYPT '91*, pp. 490–497, Springer, Berlin, Heidelberg, 1991.

[22] G. Yang and C. H. Tan, "Certificateless cryptography with KGC trust level 3," *Theoretical Computer Science*, vol. 412, no. 39, pp. 5446–5457, 2011.

[23] F. Li, W. Gao, D. Xie, and C. Tang, "Certificateless cryptography with KGC trust level 3 revisited," in *Cloud Computing and Security*, pp. 292–304, Springer, 2017.

[24] B. C. Hu, D. S. Wong, Z. Zhang, and X. Deng, "Certificateless signature: a new security model and an improved generic construction," *Designs, Codes and Cryptography*, vol. 42, no. 2, pp. 109–126, 2007.

[25] Y.-C. Chen, R. Tso, G. Horng, C.-I. Fan, and R.-H. Hsu, "Strongly secure certificateless signature: cryptanalysis and improvement of two schemes," *Journal of Information Science and Engineering*, vol. 31, no. 1, pp. 297–314, 2015.

[26] Y.-F. Tseng, C.-I. Fan, and C.-W. Chen, "Top-level secure certificateless signature scheme in the standard model," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2763–2774, 2019.

[27] P. Rastegari and W. Susilo, "On random-oracle-free top-level secure certificateless signature schemes," *The Computer Journal*, vol. 65, no. 12, pp. 3049–3061, 2022.

[28] C. Schnorr, "Efficient identification and signatures for smart cards," *Advances in Cryptology - Crypto 89: Proceedings*, , , pp. 239–251, Springer, Berlin, 1990.

[29] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *Journal of Cryptology*, vol. 13, no. 3, pp. 361–396, 2000.