Certificate-Based Parallel Key-Insulated Aggregate Signature Against Fully Chosen Key Attacks for Industrial Internet of Things

Yingzhe Hou, Hu Xiong, Xin Huang, and Saru Kumari

***Abstract*—With the emergence of the Industrial Internet of Things (IIoT), numerous operations based on smart devices contribute to producing the convenience and comfortable applications for individuals and organizations. Considering the untrusted feature of the communication channels in IIoT, it is essential to ensure the authentication and incontestableness of the messages transmitted in the IIoT. In this article, we first proposed a certificate-based parallel key-insulated aggregate signature (CB-PKIAS), which can resist the fully chosen-key attacks. Concretely, the adversary who can obtain the private keys of all signers in the system is able to forge a valid aggregate signature by using the invalid single signature. Furthermore, our scheme inherits the merits of certificate based and key insulated to avoid the certificate management problem, key-escrow problems, as well as the key exposures simultaneously. In addition, the rigorous analysis and the concrete simulation experiment demonstrated that our proposed scheme is secure under the random oracle and more suitable for the IIoT environment.**

***Index Terms*—Aggregate signature, certificate based, fully chosen-key attacks, Industrial Internet of Things (IIoT), key insulated.**

I. Introduction

IN RECENT years, the increase of digitalization has greatly facilitated the prosperity of the emerging field named Industrial Internet of Things (IIoT) [1], which dedicates to fabricate a more intelligent management system with less human intervention. By the ubiquitous smart devices, such as machinery, actuators, and sensors, it is possible for industrial companies to manage the valuable information and the interaction with each other more intelligently and efficiently [2]. As the Fig. 1 shows, different types of sensors provide opportunities for collecting data in various fields, such as temperature sensors, humidity sensors, light sensors, sound and noise sensors, motion sensors, etc. Considering the dominance of the connected sensors during the industrial platform, IIoT attracts more attention since the traditional IoT from the engineering and academic community.

Despite numerous benefits are brought by IIoT, the authenticity of data is still a critical problem that needs an appropriate solution. To address the former challenge, how to preserve the authenticity of data transmitted between the smart devices and the third party is a breakthrough. The digital signature [3] seems to be a promising approach to protect data from forging and tampering during the transmission process. In this manner, data will be signed by the signer's private key before being delivered to other devices. Afterward, this primitive enables the receiver to verify the signature to guarantee the authenticity of data reasonably. Subsequently, a series of signature protocols based on the identity (ID)-based cryptosystem (IBC) or public-key infrastructure (PKI) are gradually put forward [4]–[6].

With respect to the PKI cryptosystem [7], [8], there exists a trusted organization that can generate the certificate corresponding to the user's ID. Nevertheless, the certificate management problem cannot be ignored, which greatly reduces the applicability of the PKI-based schemes. For addressing this obstacle, the signature scheme from the IBC system (IBS) is proposed [3]. In this construction, the private-key generator (PKG) and the ID generate the private/public key of user [9]. Although the certificate management problem can be solved by replacing the certificate with the user's ID, this mechanism results in an equally important key-escrow problem since the private key can be calculated by PKG easily.

For tackling the above-mentioned key-escrow problem, the certificate-based signature schemes (CBS) [10], [11] are proposed, which both eliminate the inherent defects of schemes from PKI and IBC mechanisms. When n signatures are corresponding to n different messages of n users in the system, the aggregate signature can aggregate them into a single short signature and require to be verified only once, which allows bandwidth and computing savings. Therefore, it is essential to introduce the notion of aggregate signature and this primitive makes it suitable for environments with resource-constrained conditions. Inspired by this perspective, various types of schemes based on certificate-based aggregate signature (CBAS) are designed at a rapid speed [12], [13].

In a traditional aggregate signature protocol, the involved adversary always obtains $(n-1)$ private keys, where the number of system users is n. Nonetheless, as one of the special attack patterns, a fully chosen-key attacker can hold all the

Manuscript received July 28, 2020; revised September 16, 2020, November 14, 2020, and December 23, 2020; accepted January 26, 2021. Date of publication February 2, 2021; date of current version May 21, 2021. This work was supported in part by the Natural Science Foundation of China under Grant U1936101; and in part by the 13th Five-Year Plan of National Cryptography Development Fund for Cryptographic Theory of China under Grant MMJJ20170204. *(Corresponding author: Saru Kumari.)*

Yingzhe Hou, Hu Xiong, and Xin Huang are with the School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China.

Saru Kumari is with the Department of Mathematics, Chaudhary Charan Singh University, Meerut 250004, India (e-mail: saryusiirohi@gmail.com).

Digital Object Identifier 10.1109/JIOT.2021.3056477

2327-4662 © 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: NANJING NORMAL UNIVERSITY. Downloaded on December 07,2022 at 10:48:34 UTC from IEEE Xplore. Restrictions apply.

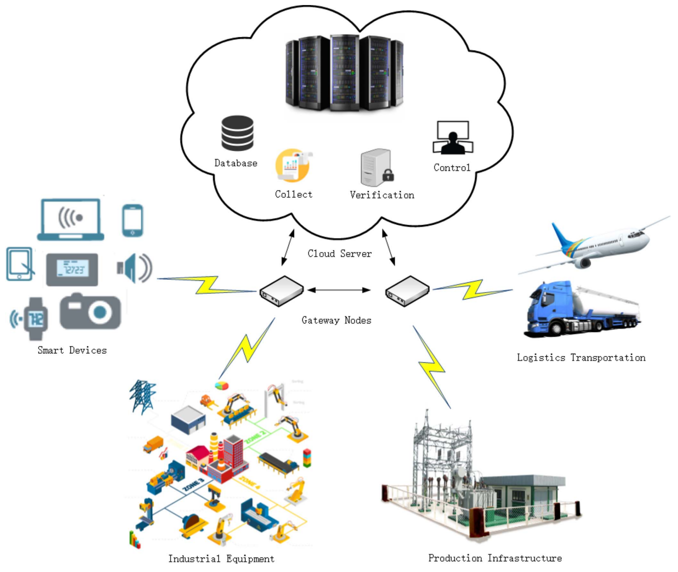


Fig. 1. Typical scenario of cloud-assisted IIoT environment.

private key and the purpose of it is to break the aggregate signature scheme's security. For example, a single signature forged by this fully chosen-key attacker is invalid, but the aggregate signature generated by the aforementioned single signature is valid. Wu *et al.* [14] introduced a certificateless aggregate signature scheme that describes the way to resist the fully chosen-key attacks.

What makes things worse, the key disclosure problem is also inevitable since the signature primitive often deployed in the insecure channel. Dodis *et al.* [15] first presented the concept of key insulated, which offered a new solution to ease this challenge. In their proposed scheme, the private key includes user's private key as well as helper key. Specifically, the user's private key remains changing to provide the signature functionality during the period, while the helper key generated by the physical device is responsible for updating the previous key. In addition, the user's public key is a constant value. Another problem encountered in [15] is that only a helper is supported, which leads to an increase in frequency of key updates, thereby resulting key exposure problem. Thus, the concept of key insulated with the functionality of parallel is proposed by Hanaoka *et al.* [16]. In this way, the decryption key is updated by two independent helper keys. This improvement not only allows us to update the decryption keys frequently, but also avoids the helper key's exposure.

Recently, Verma *et al.* [13] presented an efficient CBS scheme with compact aggregation (CB-CAS). They alleged that their CB-CAS scheme is secure. Conversely, after carefully observing their scheme, we show that the scheme in [13] fails to achieve the claimed security features and still suffers the attacks from public key replacement, the malicious KGC, the fully chosen and even an outsider. Besides, the fundamental reasons why their scheme is insecure as well as the guideline for resisting these attacks during the design of certificate-based signature are also presented. In summary, in order to resist the previous attacks and achieve the parallel key-insulated property, we proposed a certificate-based parallel key-insulated aggregate signature (CB-PKIAS) against the fully chosen-key attacks for IIoT. The concrete contributions of the proposed scheme are demonstrated as follows.

1) This article introduces the different types of forgery attacks involved in the scheme [13]. Afterward, we analyze the basic reasons for the insecurity of the CB-CAS scheme.
2) This article provides a new CB-PKIAS scheme, which can achieve the EUF-CMA [17] and resist the fully chosen-key attacks. Besides, the key exposure problem has also been addressed by offering the key-insulated property.
3) Finally, the specific experiment simulation and performance comparison are executed to reveal the practicality of our proposed scheme.

The organization is described as follows. In Section II, we give the related works. In Section III, we demonstrate the corresponding preliminaries. In Section IV, Verma *et al.*'s scheme is reviewed. In Section V, the concrete construction of CB-PKIAS is provided. In Section VI, we show the security analysis of CB-PKIAS. In Section VII, we describe the performance comparison with the existing works. Finally, several significant summaries are shown in Section VIII.

II. Related Work

After Diffie and Hellman [18] introduced the public key cryptography in 1976, many attempts are executed to put forward an actual system. Among them, many digital signature schemes based on PKI [19]–[21] are served to preserve the nonreplacement of digital documents. However, the public key of the PKI-based signature system corresponds to a specific certificate, resulting in a huge overhead caused by certificate management. For eliminating the huge overhead of the above problem, Shamir [22] proposed the ID-based digital signature scheme for the first time. Their scheme employs the public unique identifying information as the public key, thereby eliminating the certificate management overhead. On the other hand, PKG is in charge of generating the private key. Afterward, a series of ID-based signcryption schemes were proposed [23]–[26]. For reducing the signature computational overhead, a short IBS was suggested by Du and Wen [27]. Liu *et al.* [28] have proposed a novel IBS scheme. This scheme can use the multiple times with offline storage and it is suitable for wireless sensor networks. Unfortunately, the schemes based on IBS also have shortcomings. All users' private keys are produced by a PKG, so any entity's signature can be easily forged by PKG, resulting in the new problem in IBS [17].

For solving the above problems, Gentry [29] introduced the notion of certificate-based cryptography (CBC) to merge the advantages of PKI and IBC. The user generates a key pair (private/public) and obtains the certificate from a trusted authority (TA). The certificate in CBC serves as a part of the user's private key, so it can only perform operations, such as signing or decryption by using the user's certificate and private key simultaneously. Since CBC has an implicit certificate function, the need for third-party queries of the traditional PKI is eliminated, thereby simplifying complex certificate management.

Authorized licensed use limited to: NANJING NORMAL UNIVERSITY. Downloaded on December 07,2022 at 10:48:34 UTC from IEEE Xplore. Restrictions apply.

In addition, the private key in CBC is generated and retained by the user, so there is no problem of key escrow. Afterward, Kang *et al.* [10] designed the first CBS, which constructs an ID-based signature for document signing and a short signature for certification. Subsequently, a series of CBS schemes were proposed [30]–[32]. Zhou and Cui presented a CBS [33] which can resist the attack from malicious-but-passive certifier.

Unfortunately, the previous schemes' security is based on an assumption: the private key of the user is completely secured. In fact, the operations of signature are usually performed in insecure devices or environments, so the issue of signing key exposure seems inevitable. Therefore, Du *et al.* [34] put forward the key-insulated signature (KIS) into the certificate-based signcryption scheme, and then introduced the concept of certificate-based signature (CB-KIS) and the first CB-KIS scheme. In the KIS mechanism, the private key of user includes different time slices and will update in different time slices through the physical security device. In this way, the impact of key exposure is mitigated. Afterward, Xiong *et al.* [35] put forward a novel CB-KIS scheme, which is pairing-free. Li *et al.* [36] provided a CB-KIS scheme, which simplifies the certificate management through the function of key insulation. Xiong *et al.* [37] introduced a signature supporting parallel key insulated (CL-PKIS), which is following the works in [16] and [38].

In addition to solving the above problems, for reducing the overhead of data transmission, a digital signature aggregation mechanism was introduced. This mechanism was first proposed by Boneh *et al.* [39]. In this concept, n signatures of n messages are compressed to form a short signature. The corresponding messages of n signers can be confirmed by verifying the short signature, thereby reducing the total bandwidth required for transmission and the total calculation cost of the verification process. Subsequently, Liu *et al.* [12] provided a CBAS. It uses sequential aggregation, where the signer creates an aggregated signature (AS) based on the previous AS. Therefore, the aggregation is performed by each signer. Ma *et al.* [40] introduced a CB-CAS scheme, but their scheme aggregates different signatures from several signers. Verma *et al.* [41] presented a CB-CAS scheme for electronic medical monitoring the signers' number determined the signature's length. Recently, the first pairing free certificate-based compact aggregate signature was proposed by Verma *et al.* [13]. In this scheme, compact aggregation is used to create a fixed-length AS, so the final AS length will not be affected by the increase in the number of signatures. In summary, the CB-PKIAS scheme has not been presented.

III. Preliminaries

In this section, the relevant preliminaries for a better understanding are described as follows.

A. Bilinear Map

Define two groups G_1 and G_2, which regarded q as its prime order. Besides, set the bilinear map $e : G \times G \to G_T$ [42] and meet the properties as below.

1) *Bilinearity:* For $\forall m, n \in Z_q^*$, $e(mP, nP) = e(P, P)^{mn}$.
2) *Nondegeneracy:* $e(P, P) \neq 1$.
3) Computability: $\exists P, Q \in G$ that can calculate $e(P, Q)$.

B. Complexity Assumption

Computational Diffie–Hellman Problem (CDHP) [43]: After receiving the input $< P, aP, bP > \in G$, in which $a, b \in Z_q^*, P \in G$, CDHP is target to compute abP.

Computational Diffie–Hellman Assumption (CDHI) [44]: If there is no probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ with a nonnegligible advantage that can calculate CDHP, it represents that CDHI is established.

C. Outline of the CB-PKIAS

The introduced scheme contains nine algorithms and the details are shown as follows.

1) *Setup:* When receiving a security parameter k, TA produces the master secret key s and the system public parameters par.
2) *CerExtract:* When inputting par and ID_i, the TA produces the certificate $Cert_i$ and returns it to the corresponding user.
3) *UserKeyExtract:* When inputting par, ID_i, and the time period t, a user generates $US_{ID,0}$ and UY_{ID} as its initial secret key and public key, respectively. Besides, it also produces the private key (HS_0, HS_1) and public key (HY_0, HY_1) of two helpers.
4) *Update*:* When inputting par, t, and the ith helper's private key HS_i, where $i \equiv t \bmod 2$, the helper generates $UD_{ID,t}$ as the update key.
5) *Update:* When inputting par, t, $US_{ID,0}$ and $UD_{ID,t}$, a user produces the temporary signing key $US_{ID,t}$.
6) *Sign:* When inputting par, $Cert_i$, $US_{ID,t}$, and a message $m_i \in \{0, 1\}^*$, a signer generates the signature σ_i.
7) *Verify:* When inputting par, ID_i, m_i, and σ_i, any verifier can validate the signature by producing "*true*" or "*false*."
8) *Aggregate:* When inputting par, ID_i, m_i, σ_i and the public verification key Y_{ver}, the aggregator generates the aggregate signature Σ.
9) *AggVer:* When inputting par, ID_i, $UY_{ID,t}$, m_i, Σ, and the secret verification key τ, the intend verifier outputs "*true*" or "*false*."

D. Threat Model

In this scheme, the current de-facto standard model CK-adversary model has been adopted [45], which simplifies the process of formal definition. Notably, the CK model can resist neither the key compromise impersonation (KCI) attack nor preserve the weak perfect forward secrecy (wPFS) of the protocol. According to the CK threat model, the signatures are delivered in the insecure channel, in which the third party for signature verification is considered as the untrusted entity. Supposed that the cloud server is physically secured to avoid the compromise of entire system security [46]. Besides, TA is defined as the fully trusted party, which is employed to generate the system parameter and the certificate of different users. Based on the CK-adversary model, adversary $\mathcal{A}$ has the ability to delete or eavesdrop the transmitted messages during

Authorized licensed use limited to: NANJING NORMAL UNIVERSITY. Downloaded on December 07,2022 at 10:48:34 UTC from IEEE Xplore. Restrictions apply.

the open channel. In addition, $\mathcal{A}$ also can reveal the crucial credentials, especially the private key [47]. In our construction, the attacker cannot forge a signature outside the time period after getting the private key since the addition of key-insulted mechanism. Therefore, the aforementioned security can be guaranteed in this scheme.

E. Security Model of the CB-PKIAS

In this section, we demonstrate the CB-PKIAS's security model, which takes three kinds of adversaries with different abilities into consideration.

Game 1: Assume that a Type-I adversary $\mathcal{A}_1$ and a challenger $\mathcal{C}$ execute the following interactions.

1) *Setup:* This operation runs the algorithm of *Setup* to produce the master secret key s and the system public key par.
2) *Query:* In this section, $\mathcal{A}_1$ executes the related operations.
   a) *Public Key Query:* When receiving this query, $\mathcal{C}$ will deliver the public key UY_{ID} to $\mathcal{A}_1$.
   b) *Public Key Replace Query:* When receiving this query, $\mathcal{C}$ will replace UY_{ID} by UY'_{ID} and return UY'_{ID} to $\mathcal{A}_1$.
   c) *Certificate Query:* When receiving this query, $\mathcal{C}$ will return the certificate $Cert_i$ to $\mathcal{A}_1$.
   d) *Signing key Query:* When receiving this query, $\mathcal{C}$ will return the temporary secret key $US_{ID,t}$ to $\mathcal{A}_1$.
   e) *Sign Query:* When receiving this query, $\mathcal{C}$ will produce the valid signature σ.
3) *Forgery:* When accomplishing the above queries, $\mathcal{A}_1$ will generate the forged signature σ^* and satisfy the conditions as follows.
   a) $\mathcal{A}_1$ cannot execute the certificate query with ID^*.
   b) $\mathcal{A}_1$ cannot execute the signing key query with ID^*.
   c) $\mathcal{A}_1$ cannot execute the sign query with (ID^*, m^*, t^*).
   d) $\mathcal{A}_1$ can generate the forged valid signature by inputting $(par, ID^*, UY^*_{ID}, HY^*_i, HY^*_{i'}, m^*, \sigma^*_i, t^*)$.

Definition 1: Assume that there does not exist that $\mathcal{A}_1$ wins the above game, we say CB-PKIAS is EUF-CMA and can achieve the perfectly key-insulated secure.

Game 2: Assume that a type-II adversary $\mathcal{A}_2$ and $\mathcal{C}$ execute the following interactions.

1) *Setup:* This operation runs the algorithm of *Setup* to generate s and par.
2) *Query:* In this section, $\mathcal{A}_2$ executes the related operations:
   a) *Public Key Query:* When receiving this query, $\mathcal{C}$ will deliver UY_{ID} to $\mathcal{A}_2$.
   b) *Sign Query:* When receiving this query, $\mathcal{C}$ will produce the valid signature σ to $\mathcal{A}_2$.
3) *Forgery:* When accomplishing the above queries, $\mathcal{A}_2$ will generate the forged signature σ^* and satisfy the conditions as follows.
   a) $\mathcal{A}_2$ cannot execute the signing key query with the challenge identity ID^*.
   b) $\mathcal{A}_2$ cannot execute the sign query with (ID^*, m^*, t^*).
   c) $\mathcal{A}_2$ can generate the forged valid signature by inputting $(par, ID^*, UY^*_{ID}, HY^*_i, HY^*_{i'}, m^*, \sigma^*_i, t^*)$.

Definition 2: Assume that there does not exist that $\mathcal{A}_2$ wins the above game, we say CB-PKIAS is EUF-CMA and can achieve the perfectly key-insulated secure.

Game 3: Assume that a fully chosen attacker $\mathcal{A}_3$ and $\mathcal{C}$ execute the following interactions.

1) *Setup:* This operation runs the algorithm of *Setup* to generate s and par.
2) *Query:* In this section, $\mathcal{A}_3$ executes the related operations.
   a) *Signing Key Query:* When receiving this query, $\mathcal{C}$ will return the temporary secret key $US_{ID,t}$ to $\mathcal{A}_3$.
   b) *Aggver Query:* When submitting this query, $\mathcal{C}$ executes the algorithm of *AggVer* and delivers the verification result to $\mathcal{A}_3$.
3) *Forgery:* When accomplishing the above queries, $\mathcal{A}_3$ will generate the forged aggregate signature Σ and satisfy the conditions as follows.
   a) All single signatures are aggregated into the aggregate signatures Σ.
   b) The aforementioned Σ is valid.
   c) At least one signature σ'_i cannot hold the verification equation.

Definition 3: If there does not exist $\mathcal{A}_3$ belonging to the PPT adversary that wins the above game, we say that CB-PKIAS can resist the fully chosen-key attack.

Game 4: Assume that adversary $\mathcal{A}_4$ and a challenger $\mathcal{C}$ execute the following interactions.

1) *Setup:* This operation runs the algorithm of *Setup* to generate s and par. Then, $\mathcal{C}$ delivers par to $\mathcal{A}_4$.
2) *Query:* In this section, $\mathcal{A}_4$ performs a similar operation as shown in $\mathcal{A}_1$, except for the following query.
   a) *Helper Key Query:* When submitting this query, $\mathcal{C}$ will return the helper's private/public key (HS_0, HS_1, HY_0, HY_1) to $\mathcal{A}_4$.
3) *Forgery:* When accomplishing the above queries, $\mathcal{A}_4$ will generate the forged aggregate signature Σ and satisfy the conditions as follows.
   a) $\mathcal{A}_4$ cannot execute the certificate query with ID^*.
   b) $\mathcal{A}_4$ cannot execute the signing key query with ID^*.
   c) $\mathcal{A}_4$ cannot execute the sign query with (ID^*, m^*, t^*).
   d) $\mathcal{A}_4$ can generate the forged valid signature by inputting $(par, ID^*, UY^*_{ID}, HY^*_i, HY^*_{i'}, m^*, \sigma^*_i, t^*)$.

Definition 4: Assume that there does not exist that $\mathcal{A}_4$ wins the above game, we say CB-PKIAS is EUF-CMA and can achieve the strongly key-insulated secure.

Game 5: Assume that adversary $\mathcal{A}_5$ and $\mathcal{C}$ execute the following interactions.

1) *Setup:* This operation runs the algorithm of *Setup* to generate s and par. Then, $\mathcal{C}$ delivers s and par to $\mathcal{A}_5$.
2) *Query:* In this section, $\mathcal{A}_5$ performs a similar operation as shown in $\mathcal{A}_2$, except for the following query:

Authorized licensed use limited to: NANJING NORMAL UNIVERSITY. Downloaded on December 07,2022 at 10:48:34 UTC from IEEE Xplore. Restrictions apply.

a) *Helper Key Query:* When submitting this query, $\mathcal{C}$ will return the helper's private/public key (HS_0, HS_1, HY_0, HY_1) to $\mathcal{A}_5$.

3) *Forgery:* When accomplishing the above queries, $\mathcal{A}_5$ will generate the forged signature σ^* and satisfy the conditions as follows:
   a) $\mathcal{A}_5$ cannot execute the signing key query with ID^*;
   b) $\mathcal{A}_5$ cannot execute the sign query with (ID^*, m^*, t^*);
   c) $\mathcal{A}_5$ can generate the forged valid signature by inputting $(par, ID^*, UY^*_{ID}, HY^*_i, HY^*_{i'}, m^*, \sigma^*_i, t^*)$.

Definition 5: Assume that there does not exist that $\mathcal{A}_5$ wins the above game, we say CB-PKIAS is EUF-CMA and can achieve the strongly key-insulated secure.

IV. Review of Verma *et al.*'s CB-CAS Scheme

A. Overview of Verma et al.'s CB-CAS Scheme

We first give an overview of Verma *et al.*'s protocol as follows.

1) *Setup (k):* Given the security parameter λ, a TA executes this algorithm to output the system parameters $par = (q, H_0, H_1, G_T, P, P_{pub}, \Delta)$ and a master secret key s. The details are shown as follows.
   a) TA first chooses a cyclic additive group G_T of order q with generator P. It also picks $H_0 : \{0,1\}^* \times G_T \rightarrow Z_q^*$ and $H_1 : \{0,1\}^* \times G_T \times \{0,1\}^* \times \{0,1\} \rightarrow Z_q^*$ as two hash functions.
   b) Furthermore, TA randomly picks $s \in Z_q^*$. Then, calculates its public key $P_{pub} = sP$.
   c) Finally, TA picks $\Delta \in \{0,1\}^*$ as the state information and publishes: $par = (q, H_0, H_1, G_T, P, P_{pub}, \Delta)$.

2) *UserKeyExtract (par):* Given par, the user with identity ID_i performs this algorithm. Concretely, it picks $x_i \in Z_q^*$ and calculates $Y_i = x_iP$ as the private/public key.

3) *CerExtract (par, Y_i, ID_i):* Given par, Y_i and ID_i, this algorithm is executed by TA to generate the certificate $Cert_i$. Specifically, TA does the following operations.
   a) Pick $w_i \in Z_q^*$ and compute $W_i = w_iP$.
   b) Compute $c_i = w_i + sH_0(\mathrm{ID}_i||Y_i)$.
   c) Return $Cert_i = (W_i, c_i)$ to the requesting signer.

4) *Sign $(par, Cert_i, x_i, m_i)$:* Given par, $Cert_i$, x_i and the message $m_i \in \{0,1\}^*$, a signer runs this algorithm to output the signature σ. To be specific, the signer does the following operations.
   a) Check the certificate authenticity through $c_iP \overset{?}{=} U_i + H_0(\mathrm{ID}_i||Y_i)P_{pub}$. If the equation holds, $Cert_i$ is valid for further operation.
   b) Pick $k_i \in Z_q^*$ and calculate $U_i = W_i + k_iP$.
   c) Calculate $v_i = k_i + c_i + x_iH_1(m_i||Y_i||\mathrm{ID}_i||\Delta)$ and send $\sigma_i = (U_i, v_i)$ back to the aggregator.

5) *Verify (par, ID_i, σ_i):* Given par, $Cert_i$, ID_i, and σ_i, this algorithm is executed by a receiver. If $v_iP = U_i + H_0(\mathrm{ID}_i||Y_i)P_{pub} + H_1(m_i||Y_i||\mathrm{ID}_i||\Delta)Y_i$ holds, the algorithm will generate "*true*." If not, generate "*false*."

6) *Aggregate:* This algorithm is ran by an aggregator to generate an aggregation signature on $(m_1, m_2, m_3 \cdots m_n)$. The aggregator first checks whether $v_iP = U_i + H_0(\mathrm{ID}_i||Y_i)P_{pub} + H_1(m_i||Y_i||\mathrm{ID}_i||\Delta)Y_i$ holds or not. If it holds the verification, the aggregator computes $U = \sum_{i=1}^n U_i$ and $v = \sum_{i=1}^n v_i$. Finally, this algorithm generates (R, z) as the aggregation signature.

7) *AggVer:* The validity of the aggregation signature is checked by a receiver. If $vP = U + (\sum_{i=1}^n H_0(\mathrm{ID}_i||Y_i))P_{pub} + \sum_{i=1}^n H_1(m_i||Y_i||\mathrm{ID}_i||\Delta)Y_i$ holds, this algorithm outputs "*true*"; otherwise, it outputs "*false*."

B. Weakness of Verma et al.'s Scheme

In this section, four types of forgery attacks are given to demonstrate the defects of Verma *et al.*'s protocol. Concretely, attack I is performed by the type I adversary, who executes the attack of public-key replacement [48]. Attack II is mounted by the type II adversary, which is a malicious KGC [49]. Attack III is launched by any outside attacker without replacing the user's public key and accessing s. Attack IV is executed by a fully chosen attacker. Furthermore, if an attacker has the ability to forge a single signature, it can forge an aggregation signature simultaneously.

1) Attack I (Attack From Type I Adversary): If a type I adversary $\mathcal{A}_1$ can forge a valid signature σ^* on any message m_i representing the user with public key Y_i and identity ID_i, $\mathcal{A}_1$ is allowed to produce the forged signature by replacing the current public key Y_i as follows.

1) Randomly choose $x_i^* \in Z_q^*$ and set $Y_i^* = x_i^*P$.
2) Pick $U_i \in Z_q^*$ and set $U_i^* = U_iP - H_0(\mathrm{ID}_i||Y_i^*)P_{pub}$.
3) Set $v_i^* = U_i + x_i^*H_1(m_i||Y_i^*||\mathrm{ID}_i||\Delta)$.
4) Generate $\sigma^* = (U_i^*, v_i^*)$ as the forged signature on m_i.

It is easy to observe that σ^* is valid with the condition of replacing Y_i with Y_i^*. The correctness of σ^* is shown as follows:

$$\begin{aligned}
&U_i^* + H_0(\mathrm{ID}_i||Y_i^*)P_{pub} + H_1(m_i||Y_i^*||\mathrm{ID}_i||\Delta)Y_i^* \\
&\quad = U_iP - H_0(\mathrm{ID}_i||Y_i^*)P_{pub} + H_0(\mathrm{ID}_i||Y_i^*)P_{pub} \\
&\qquad + H_1(m_i||Y_i^*||\mathrm{ID}_i||\Delta)Y_i^* \\
&\quad = U_iP + H_1(m_i||Y_i^*||\mathrm{ID}_i||\Delta)Y_i^* \\
&\quad = U_iP + x_i^*H_1(m_i||Y_i^*||\mathrm{ID}_i||\Delta)P \\
&\quad = v_i^*P.
\end{aligned}$$

The essential reason about this attack is due to the fact that U_i and $H_0(\mathrm{ID}_i||Y_i)P_{pub}$ involved in the verification are independent of each other. In this scheme, U_i can be deliberately calculated to cancel $H_0(\mathrm{ID}_i||Y_i)P_{pub}$ and, thus, the signature could be produced without accessing s by the type I adversary.

2) Attack II (Attack From Type II Adversary): If a type II adversary $\mathcal{A}_2$ attempts to forge σ^* on any message m_i for the user with public key Y_i and identity ID_i, $\mathcal{A}_2$ has the ability to generate a forged signature without knowing user's secret key as follows.

1) Randomly select $U_i \in Z_q^*$ and set $U_i^* = U_iP - H_1(m_i||Y_i||\mathrm{ID}_i||\Delta)Y_i$.
2) Set $v_i^* = U_i + sH_0(\mathrm{ID}_i||Y_i)$.
3) Generate $\sigma^* = (U_i^*, v_i^*)$ as the forged signature on m_i.

Authorized licensed use limited to: NANJING NORMAL UNIVERSITY. Downloaded on December 07,2022 at 10:48:34 UTC from IEEE Xplore. Restrictions apply.

Obviously, the forged signature σ^* is valid. We demonstrate the concrete steps of correctness as follows:

$$\begin{aligned}
&U_i^* + H_0(\mathrm{ID}_i||Y_i)P_{pub} + H_1(m_i||Y_i||\mathrm{ID}_i||\Delta)Y_i\\
&\quad = U_iP - H_1(m_i||Y_i||\mathrm{ID}_i||\Delta)Y_i + H_0(\mathrm{ID}_i||Y_i)P_{pub}\\
&\qquad + H_1(m_i||Y_i||\mathrm{ID}_i||\Delta)Y_i\\
&\quad = U_iP + H_0(\mathrm{ID}_i||Y_i)P_{pub}\\
&\quad = U_iP + sH_0(\mathrm{ID}_i||Y_i)P\\
&\quad = v_i^*P.
\end{aligned}$$

Similar to the previous attack, the reason of our attack depends on the U_i and $H_1(m_i||Y_i||\mathrm{ID}_i||\Delta)Y_i$ in the verification are independent of each other. Thus, U_i is able to be calculated to cancel $H_1(m_i||Y_i||\mathrm{ID}_i||\Delta)Y_i$ and the signature can be forged successfully by the type II adversary without the knowledge of x.

3) Attack III (Attack From Anyone): Different from the above-mentioned two attacks, any outside attacker who neither replaces the public key nor accesses the master secret key could be considered as a legitimate user to forge σ^*. Specifically, this attacker is able to generate a valid signature with public key Y_i and identity ID_i as follows.

1) Randomly select $U_i \in Z_q^*$ and set $U_i^* = U_iP - H_0(\mathrm{ID}_i||Y_i)P_{pub} - H_1(m_i||Y_i||\mathrm{ID}_i||\Delta)Y_i$.
2) Set $v_i^* = U_i$.
3) Generate $\sigma^* = (U_i^*, v_i^*)$ as the forged signature.

We can observe that the forged signature σ^* is valid. The consistency can be verified duo to

$$\begin{aligned}
&U_i^* + H_0(\mathrm{ID}_i||Y_i)P_{pub} + H_1(m_i||Y_i||\mathrm{ID}_i||\Delta)Y_i\\
&\quad = U_iP - H_0(\mathrm{ID}_i||Y_i)P_{pub} - H_1(m_i||Y_i||\mathrm{ID}_i||\Delta)Y_i\\
&\qquad + H_0(\mathrm{ID}_i||Y_i)P_{pub} + H_1(m_i||Y_i||\mathrm{ID}_i||\Delta)Y_i\\
&\quad = U_iP = v_i^*P.
\end{aligned}$$

In this case, the inherent reason about this security flaw is that U_i is independent of both $H_0(\mathrm{ID}_i||Y_i)P_{pub}$ and $H_1(m_i||Y_i||\mathrm{ID}_i||\Delta)Y_i$ in the verification. This attacker can set a mendacious value U_i^* to offset $H_0(\mathrm{ID}_i||Y_i)P_{pub}$ and $H_1(m_i||Y_i||\mathrm{ID}_i||\Delta)Y_i$ simultaneously. Therefore, anyone can generate the signature without replacing the public key and accessing the master secret key.

4) Attack IV (Fully Chosen-Key Attack): Assume that a fully chosen-key attacker intends to forge aggregate signature Σ^* on m_1 and m_2. Some operations are executed in the sign algorithm.

1) Randomly pick $k_1, k_2 \in Z_q^*$ and calculate $U_1 = W_1 + k_1P$ and $U_2 = W_2 + k_2P$.
2) Randomly pick $e \in Z_q^*$, calculate
$$\begin{aligned}
v_1 &= k_1 + c_1 + x_1H_1(m_1||Y_1||\mathrm{ID}_1||\Delta) + e\\
v_2 &= k_2 + c_2 + x_2H_1(m_2||Y_2||\mathrm{ID}_2||\Delta) - e.
\end{aligned}$$
3) Calculate $v = v_1 + v_2$.
4) Generate $\sigma_1 = (U_1, v_1)$ and $\sigma_2 = (U_2, v_2)$ as the forged signature on m_1 and m_1. Generate $\Sigma^* = (U_1, U_1, v)$ as the aggregate signature.

We can observe that the forged signature Σ^* is valid, while the single signature σ_1 and σ_1 is invalid. The correctness of Σ^* is shown as follows:

$$\begin{aligned}
&U_1 + H_0(\mathrm{ID}_1||Y_1)P_{pub} + H_1(m_1||Y_1||\mathrm{ID}_1||\Delta)Y_1\\
&\quad + U_2 + H_0(\mathrm{ID}_2||Y_2)P_{pub} + H_1(m_2||Y_2||\mathrm{ID}_2||\Delta)Y_2\\
&= \sum_{i=1}^{2} U_i + \left(\sum_{i=1}^{2} H_0(\mathrm{ID}_i||Y_i)\right)P_{pub} + \sum_{i=1}^{2} H_1(m_i||Y_i||\mathrm{ID}_i||\Delta)Y_i\\
&= (v_1 + v_2)\cdot P\\
&= v \cdot P.
\end{aligned}$$

The essential reason about this attack is that the single signature cannot be verified in time. In this scheme, v_i can be deliberately calculated by the fully chosen-key attacker.

V. Proposed CB-PKIAS Scheme

A. Construction

The proposed CB-PKIAS includes nine different algorithms, which are described as follows. The specific process of scheme is shown in Fig. 2.

1) *Setup (k):* Randomly pick $\tau \in Z_q^*$ as the secret verification key of the intended verifier and then compute the corresponding public verification key as $Y_{ver} = \tau P$. Given λ, TA runs the following operations to generate the system parameters par and the master secret key s. The details are shown as follows.
   a) Pick two cyclic additive group G and G_T of order q. Pick $H_0 : \{0,1\}^* \to G$, $H_1 : \{0,1\}^* \times G \to G$, $H_2 : \{0,1\}^{*3} \times G^4 \to Z_q^*$, $H_3 : G_T^n \to \{0,1\}^*$, and $H_4 : \{0,1\}^* \times G \times Z_q^* \to Z_q^*$ as five hash functions. Then, Choose d_1 and d_2 as two bit strings and their length is l. Calculate $P = H_0(d_1)$ and $Q = H_0(d_2)$.
   b) Pick $s \in Z_q^*$ and calculate its public key $T_{pub} = sP$.
   c) Pick $\Delta \in \{0,1\}^*$ as the state information and publishes: $par = (l, d_1, d_2, H_0, H_1, H_2, H_3, H_4, G, G_T, P, Q, \Delta, T_{pub})$.
2) *CerExtract (par, ID_i):* Given par and ID_i, this algorithm is executed by TA for generating the certificate $Cert_i$. Specifically, TA does the following steps.
   a) Pick $w_i \in Z_q^*$ and compute $W_i = w_iP$.
   b) Compute $\Psi_i = H_1(\mathrm{ID}_i||W_i)$.
   c) Compute $\Lambda_i = w_iQ + s\Psi_i$.
   d) Return $Cert_i = (W_i, \Lambda_i)$ to the requesting signer.
3) *UserKeyExtract (par, t):* Given par and the time period t, a user will perform this algorithm to calculate $US_{ID,0}$ and UY_{ID} as its initial secret/public key. Besides, (HS_0, HS_1) and (HY_0, HY_1) of two helpers also be calculated.
   a) Select the secret value $x_{ID} \in Z_q^*$ and calculate $UY_{ID} = x_{ID}\cdot P$.
   b) Choose $c_0, c_1 \in Z_q^*$, set $HS_0 = c_0$ and $HS_1 = c_1$, and calculate $HY_0 = c_0\cdot P$, $HY_1 = c_1\cdot P$. Afterward, the user delivers (HS_0, HS_1) to the helper and removes them from user.
   c) Calculate $f_{ID,-2} = H_4(\mathrm{ID}_i, UY_{ID}, -2)$ and $f_{ID,-1} = H_4(\mathrm{ID}_i, UY_{ID}, -1)$. Finally, calculate the initial secret key $US_{ID,0} = f_{ID,-2}\cdot c_0 + f_{ID,-1}\cdot(c_1 + x_{ID})$.

Authorized licensed use limited to: NANJING NORMAL UNIVERSITY. Downloaded on December 07,2022 at 10:48:34 UTC from IEEE Xplore. Restrictions apply.

4) *Update*(par, t, c_i):* Given *par*, *t*, and the private key c_i of the *i*th helper, where $i' \equiv (t-1) \bmod 2$. Finally, the update key is $UD_{ID,t} = c_i' \cdot (f_{ID,t-1} - f_{ID,t-3})$.
5) *Update(par, t, $US_{ID,0}$, $UD_{ID,t}$):* Given *par*, *t*, $US_{ID,0}$, and $UD_{ID,t}$, the user with identity ID_i can calculate the temporary signing key $US_{ID,t} = US_{ID,t-1} + UD_{ID,t} + x_{ID} \cdot (f_{ID,t-1} - f_{ID,t-2})$, where $i \equiv t \bmod 2$ and $i' \equiv (t-1) \bmod 2$. Therefore, calculate $US_{ID,t} = f_{ID,t-2} \cdot c_i + f_{ID,t-1} \cdot (c_{i'} + x_{ID})$.
6) *Sign (par, $Cert_i$, $US_{ID,t}$, m_i):* Given *par*, $Cert_i$, $US_{ID,t}$, and a message $m_i \in \{0,1\}^*$, the signer with identity ID_i runs this algorithm to output the signature σ_i. To be specific, the signer does the following steps.
   a) Pick $k_i \in Z_q^*$ and calculate $K_i = k_i P$.
   b) Calculate $\xi_i = H_2(\mathrm{ID}_i || m_i || \Delta || UY_{ID} || T_{pub} || W_i || K_i)$.
   c) Calculate $R_i = \Lambda_i + \xi_i \cdot k_i \cdot T_{pub} + \xi_i \cdot US_{ID,t} \cdot Q$ and send $\sigma_i = (W_i, K_i, R_i)$ back to the aggregator.
7) *Verify(par, ID_i, UY_{ID}, HY_i, $HY_{i'}$, m_i, σ_i):* Given *par*, ID_i, UY_{ID}, HY_i, $HY_{i'}$, m_i, and σ_i, this algorithm can be executed by any user via the following steps.
   a) Compute $\Psi_i = H_1(\mathrm{ID}_i || W_i)$ and $\xi_i = H_2(\mathrm{ID}_i || m_i || \Delta \; || UY_{ID} || T_{pub} || W_i || K_i)$.
   b) Check the following equation:

$$\begin{aligned} e(R_i, P) = {} & e(\Psi_i + \xi_i \cdot K_i, T_{pub}) \\ & \times e(W_i + \xi_i \cdot [f_{ID,t-2} HY_i \\ & + f_{ID,t-1}(HY_{i'} + UY_{ID})], Q). \end{aligned}$$

8) *Aggregate(par, $(ID_i, UY_{ID}, m_i, \sigma_i) | i = 1, \ldots, n$, Y_{ver}):* Given *par*, ID_i, UY_{ID}, m_i, σ_i, and the public verification key $Y_{ver} = \tau P$, an aggregator can execute the following operations.
   a) Compute $R = \sum_{i=1}^{n} R_i$.
   b) Compute $\gamma = H_3(e(R_1, Y_{ver}) || \cdots || e(R_n, Y_{ver}))$.
   c) Generate the aggregate signature $\Sigma = (W_1, \ldots, W_n, K_1, \ldots, K_n, R, \gamma)$.
9) *AggVer(par, $(ID_i$, UY_{ID}, HY_i, $HY_{i'}$, $m_i) | i = 1, \ldots, n$, Σ, τ):* Given *par*, ID_i, UY_{ID}, HY_i, $HY_{i'}$, m_i, Σ, and the secret verification key τ, the intend verifier executes the following algorithms.
   a) Compute $\Psi_i = H_1(\mathrm{ID}_i || W_i)$, $\xi_i = H_2(\mathrm{ID}_i || m_i || \Delta || UY_{ID} || T_{pub} || W_i || K_i)$ for $i = 1, \ldots, n$.
   b) Compute $\Theta_i = f_{ID,t-2} HY_i + f_{ID,t-1}(HY_{i'} + UY_{ID})$.
   c) Check whether the equations hold or not

$$\begin{aligned} e(R, P) = {} & e\left(\sum_{i=1}^{n} (\Psi_i + \xi_i \cdot K_i), T_{pub}\right) \\ & \times e\left(\sum_{i=1}^{n} (W_i + \xi_i \cdot \Theta_i), Q\right) \end{aligned}$$

$$\gamma = H_3 \begin{pmatrix} e(\Psi_1 + \xi_1 \cdot K_1, \tau \cdot T_{pub}) \\ \cdot e(W_1 + \xi_1 \cdot \Theta_1, \tau \cdot Q) || \cdots || \\ e(\Psi_n + \xi_n \cdot K_n, \tau \cdot T_{pub}) \\ \cdot e(W_n + \xi_n \cdot \Theta_n, \tau \cdot Q) \end{pmatrix}.$$

   If the above-mentioned equations hold, this algorithm generates "*true*"; otherwise, it generates "*false*."

B. Design Philosophy

Considering the above reasons why the CB-CAS scheme in [13] is insecure, we provide the guideline to resist these attacks in the construction of certificate-based signature. The effective solution is to change the input of hash functions and the following steps are given to demonstrate the feasibility.

The fatal reason about attack I is that the master private key s is not embedded in the suitable position so that the adversary $\mathcal{A}_1$ can forge a valid signature by canceling $H_0(\mathrm{ID}_i || Y_i) P_{pub}$ directly. According to this reason, the simple way to resist this attack is adding U_i into $H_0(\mathrm{ID}_i || Y_i)$, where $U_i = U_i P, U_i \in Z_q^*$. Because U_i is randomly selected, the value of U_i and $H_0(\mathrm{ID}_i || Y_i || U_i)$ also changed accordingly. Therefore, $\mathcal{A}_1$ cannot forge a signature without knowing the accurate value of $H_0(\mathrm{ID}_i || Y_i || U_i)$. Similar to the previous analysis, U_i is added into $H_1(m_i || Y_i || \mathrm{ID}_i || \Delta)$ to resist the adversary from attack II. Besides, in order to resist the attack III, we add U_i into $H_0(\mathrm{ID}_i || Y_i)$ and $H_1(m_i || Y_i || \mathrm{ID}_i || \Delta)$ simultaneously. Finally, the intender verifier also be given to resist the attack IV. Through the methods of analysis, it is desirable to construct a secure scheme to resist the aforementioned four kinds of attacks.

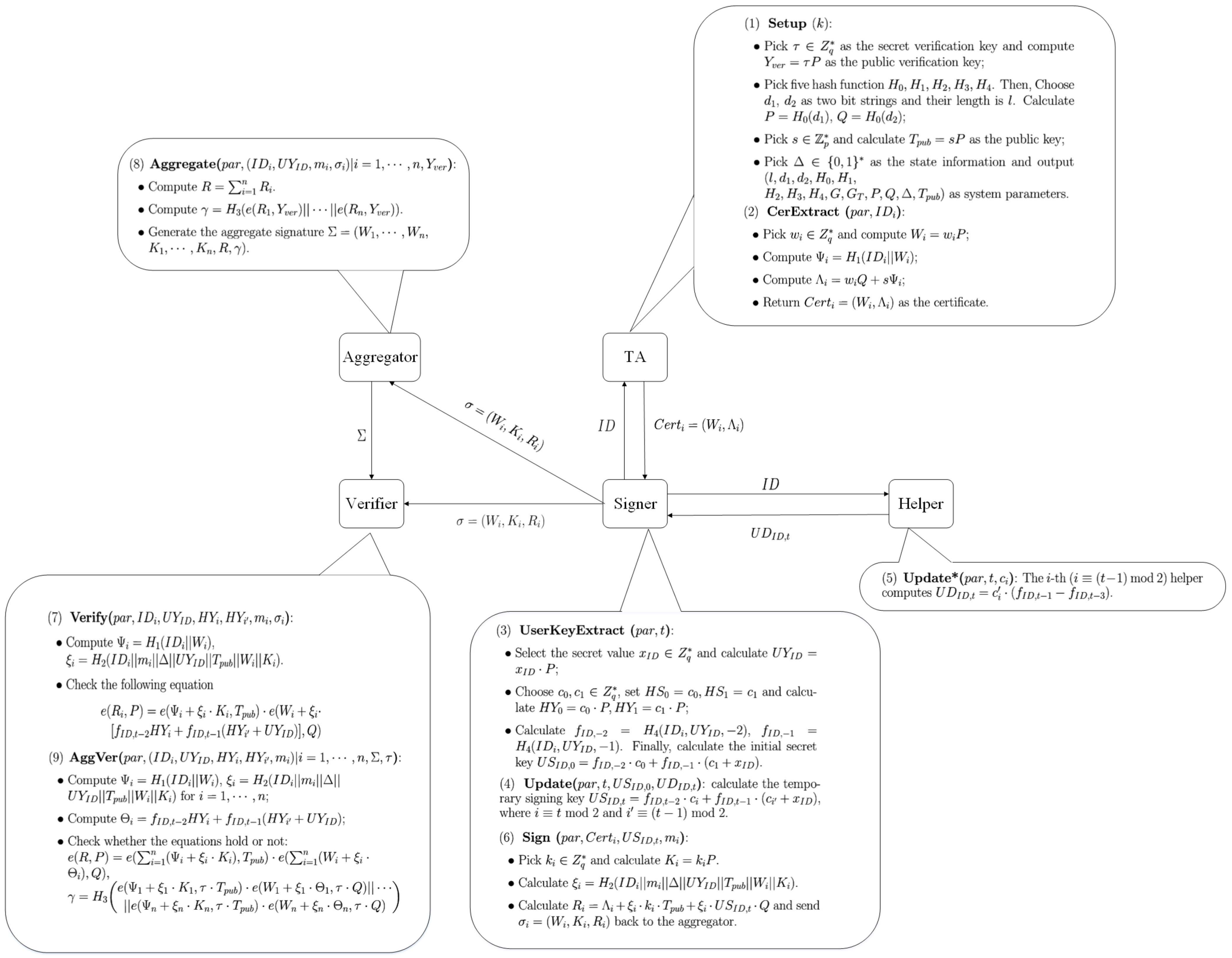
VI. Security Proof

Theorem 1: The introduced CB-PKIAS is EUF-CMA under the attack, which launched by the type I adversary.

Proof: Given (P, aP, bP) as the input of CDHP, the target of $\mathcal{C}$ is to compute abP. This process is completed by the interaction between $\mathcal{A}_1$ and $\mathcal{C}$ as follows.

1) *Setup:* $\mathcal{C}$ first picks two random bit strings d_1 and d_2 of length l, then it selects $t \in Z_q^*$ and calculates $Q = tP$. Besides, $\mathcal{C}$ sets $T_{pub} = aP$. Finally, this algorithm generates the master public key $par = (l, d_1, d_2, H_0, H_1, H_2, H_3, H_4, G, G_T, P, Q, \Delta, T_{pub})$. $\mathcal{C}$ maintains the lists $L_0, L_1, L_2, L_3, L_4, L_{pk}$, and L_t.
2) *Query:* In this section, $\mathcal{A}_1$ does the following steps.
   a) *H_0 Query:* When receiving this query on $d \in (d_1, d_2)$, $\mathcal{C}$ first checks if L_0 includes d. If so, $\mathcal{C}$ returns the stored outcome to $\mathcal{A}_1$. Otherwise, $\mathcal{C}$ picks $D_d \in G$, then it adds (d, D_d) into L_0 and returns D_d to $\mathcal{A}_1$.
   b) *H_1 Query:* When receiving this query on (ID_i, W_i), $\mathcal{C}$ first checks if L_1 includes (ID_i, W_i). If it exists, $\mathcal{C}$ returns the stored outcome to $\mathcal{A}_1$. Otherwise, $\mathcal{C}$ does the following steps.
      i) If $\mathrm{ID}_i \neq ID^*$, $\mathcal{C}$ picks $v \in Z_q^*$ and calculates $\Psi_i = vP$, then it adds $(\mathrm{ID}_i, W_i, \Psi_i)$ into L_1 and returns Ψ_i to $\mathcal{A}_1$.
      ii) If $\mathrm{ID}_i = ID^*$, $\mathcal{C}$ sets $\Psi_i = bP$, then it adds $(\mathrm{ID}_i, W_i, v, \Psi_i)$ into L_1 and returns Ψ_i to $\mathcal{A}_1$.
   c) *H_2 Query:* When receiving this query on $(\mathrm{ID}_i, m_i, \Delta, \; UY_{ID}, T_{pub}, W_i, K_i)$, $\mathcal{C}$ first checks if L_2 includes them. If yes, $\mathcal{C}$ returns the stored outcome to $\mathcal{A}_1$. Otherwise, $\mathcal{C}$ picks $\xi_i \in Z_q^*$, then it adds $(\mathrm{ID}_i, m_i, \Delta, UY_{ID}, T_{pub}, W_i, K_i, \xi_i)$ into L_2 and returns ξ_i to $\mathcal{A}_1$.

Authorized licensed use limited to: NANJING NORMAL UNIVERSITY. Downloaded on December 07,2022 at 10:48:34 UTC from IEEE Xplore. Restrictions apply.


Aggregator
TA
Verifier
Signer
Helper
Σ
$\sigma = (W_i, K_i, R_i)$
ID
$Cert_i = (W_i, \Lambda_i)$
$\sigma = (W_i, K_i, R_i)$
ID
$UD_{ID,t}$

(1) **Setup** (k):

- Pick $\tau \in Z_q^*$ as the secret verification key and compute $Y_{ver} = \tau P$ as the public verification key;
- Pick five hash function H_0, H_1, H_2, H_3, H_4. Then, Choose d_1, d_2 as two bit strings and their length is l. Calculate $P = H_0(d_1)$, $Q = H_0(d_2)$;
- Pick $s \in \mathbb{Z}_p^*$ and calculate $T_{pub} = sP$ as the public key;
- Pick $\Delta \in \{0,1\}^*$ as the state information and output $(l, d_1, d_2, H_0, H_1,$ $H_2, H_3, H_4, G, G_T, P, Q, \Delta, T_{pub})$ as system parameters.

(2) **CerExtract** (par, ID_i):

- Pick $w_i \in Z_q^*$ and compute $W_i = w_i P$;
- Compute $\Psi_i = H_1(ID_i||W_i)$;
- Compute $\Lambda_i = w_i Q + s\Psi_i$;
- Return $Cert_i = (W_i, \Lambda_i)$ as the certificate.

(8) **Aggregate**$(par, (ID_i, UY_{ID}, m_i, \sigma_i)|i = 1, \cdots, n, Y_{ver})$:

- Compute $R = \sum_{i=1}^{n} R_i$.
- Compute $\gamma = H_3(e(R_1, Y_{ver})|| \cdots ||e(R_n, Y_{ver}))$.
- Generate the aggregate signature $\Sigma = (W_1, \cdots, W_n,$ $K_1, \cdots, K_n, R, \gamma)$.

(5) **Update***(par, t, c_i): The i-th $(i \equiv (t-1) \bmod 2)$ helper computes $UD_{ID,t} = c_i' \cdot (f_{ID,t-1} - f_{ID,t-3})$.

(7) **Verify**$(par, ID_i, UY_{ID}, HY_i, HY_{i'}, m_i, \sigma_i)$:

- Compute $\Psi_i = H_1(ID_i||W_i)$, $\xi_i = H_2(ID_i||m_i||\Delta||UY_{ID}||T_{pub}||W_i||K_i)$.
- Check the following equation

$$e(R_i, P) = e(\Psi_i + \xi_i \cdot K_i, T_{pub}) \cdot e(W_i + \xi_i \cdot [f_{ID,t-2}HY_i + f_{ID,t-1}(HY_{i'} + UY_{ID})], Q)$$

(9) **AggVer**$(par, (ID_i, UY_{ID}, HY_i, HY_{i'}, m_i)|i = 1, \cdots, n, \Sigma, \tau)$:

- Compute $\Psi_i = H_1(ID_i||W_i)$, $\xi_i = H_2(ID_i||m_i||\Delta||$ $UY_{ID}||T_{pub}||W_i||K_i)$ for $i = 1, \cdots, n$;
- Compute $\Theta_i = f_{ID,t-2}HY_i + f_{ID,t-1}(HY_{i'} + UY_{ID})$;
- Check whether the equations hold or not:
$e(R, P) = e(\sum_{i=1}^{n}(\Psi_i + \xi_i \cdot K_i), T_{pub}) \cdot e(\sum_{i=1}^{n}(W_i + \xi_i \cdot \Theta_i), Q)$,
$$\gamma = H_3\begin{pmatrix} e(\Psi_1 + \xi_1 \cdot K_1, \tau \cdot T_{pub}) \cdot e(W_1 + \xi_1 \cdot \Theta_1, \tau \cdot Q)|| \cdots \\ ||e(\Psi_n + \xi_n \cdot K_n, \tau \cdot T_{pub}) \cdot e(W_n + \xi_n \cdot \Theta_n, \tau \cdot Q) \end{pmatrix}$$

(3) **UserKeyExtract** (par, t):

- Select the secret value $x_{ID} \in Z_q^*$ and calculate $UY_{ID} = x_{ID} \cdot P$;
- Choose $c_0, c_1 \in Z_q^*$, set $HS_0 = c_0, HS_1 = c_1$ and calculate $HY_0 = c_0 \cdot P, HY_1 = c_1 \cdot P$;
- Calculate $f_{ID,-2} = H_4(ID_i, UY_{ID}, -2)$, $f_{ID,-1} = H_4(ID_i, UY_{ID}, -1)$. Finally, calculate the initial secret key $US_{ID,0} = f_{ID,-2} \cdot c_0 + f_{ID,-1} \cdot (c_1 + x_{ID})$.

(4) **Update**$(par, t, US_{ID,0}, UD_{ID,t})$: calculate the temporary signing key $US_{ID,t} = f_{ID,t-2} \cdot c_i + f_{ID,t-1} \cdot (c_{i'} + x_{ID})$, where $i \equiv t \bmod 2$ and $i' \equiv (t-1) \bmod 2$.

(6) **Sign** $(par, Cert_i, US_{ID,t}, m_i)$:

- Pick $k_i \in Z_q^*$ and calculate $K_i = k_i P$.
- Calculate $\xi_i = H_2(ID_i||m_i||\Delta||UY_{ID}||T_{pub}||W_i||K_i)$.
- Calculate $R_i = \Lambda_i + \xi_i \cdot k_i \cdot T_{pub} + \xi_i \cdot US_{ID,t} \cdot Q$ and send $\sigma_i = (W_i, K_i, R_i)$ back to the aggregator.

Fig. 2. Process of CB-PKIAS scheme.

d) *H_3 Query:* When receiving this query on (R_i, Y_{ver}), $\mathcal{C}$ first checks if L_3 includes them. If yes, $\mathcal{C}$ returns the stored outcome to $\mathcal{A}_1$. Otherwise, $\mathcal{C}$ picks $j_i \in Z_q^*$, then it adds (R_i, Y_{ver}, j_i) into L_3 and returns j_i to $\mathcal{A}_1$.

e) *H_4 Query:* When receiving this query on $(\text{ID}_i, UY_{ID}, -2)$ or $(\text{ID}_i, UY_{ID}, -1)$, $\mathcal{C}$ first checks if L_4 includes them. If yes, $\mathcal{C}$ returns the stored outcome to $\mathcal{A}_1$. Otherwise, $\mathcal{C}$ picks $f \in Z_q^*$ and adds it into L_4. Finally, $\mathcal{C}$ returns f to $\mathcal{A}_1$.

f) *Public Key Query:* Define that L_{pk} stores the tuple structure $(\text{ID}_i, x_{ID}, UY_{ID}, x'_{ID}, UY'_{ID})$. When receiving this query on ID_i, $\mathcal{C}$ does the following steps.

i) If this tuple does not exist, $\mathcal{C}$ executes the algorithm of *UserKeyExtract* to output $UY_{ID} = x_{ID}P$. Finally, $\mathcal{C}$ inserts $(\text{ID}_i, x_{ID}, UY_{ID}, -, -)$ into L_{pk} and sends UY_{ID} to $\mathcal{A}_1$.

ii) If this tuple does exist, $\mathcal{C}$ sends UY_{ID} to $\mathcal{A}_1$.

iii) If this tuple does exist and the public key UY_{ID} has been replaced with UY'_{ID}. $\mathcal{C}$ sends UY'_{ID} to $\mathcal{A}_1$.

g) *Public-Key Replace Query:* When receiving this query on ID_i, $\mathcal{C}$ does the following steps.

i) If this tuple does not exist in L_{pk}, $\mathcal{C}$ executes the algorithm of *UserKeyExtract* to output $UY_{ID} = x_{ID}P$. Finally, $\mathcal{C}$ inserts $(\text{ID}_i, x_{ID}, UY_{ID}, x'_{ID}, UY'_{ID})$ into L_{pk}.

ii) If this tuple does exist, $\mathcal{C}$ replaces UY_{ID} with UY'_{ID} and sets $x_{ID} = \perp$. Finally, $\mathcal{C}$ updates the tuple with $(\perp, UY'_{ID})$.

h) *Certificate Query:* When receiving this query on ID_i, $\mathcal{C}$ does the following steps.

i) If $\text{ID}_i \neq ID^*$, $\mathcal{C}$ picks $w_i, v \in Z_q^*$ and calculates $W_i = w_iP$, $\Lambda_i = w_iQ + vT_{pub}$. Finally, $\mathcal{C}$ returns $Cert_i = (W_i, \Lambda_i)$ to $\mathcal{A}_1$.

ii) If $\text{ID}_i = ID^*$, $\mathcal{C}$ aborts.

i) *Signing Key Query:* When receiving this query on (ID_i, t), $\mathcal{C}$ maintains the list $L_t = \{\text{ID}_i, HS_0, HS_1, HY_0, HY_1\}$. Afterward, $\mathcal{C}$ checks if ID_i exists in L_t. If so, $\mathcal{C}$ randomly chooses $tc_0, tc_1 \in Z_q^*$, then calculates $HS_0 = tc_0, HS_1 = tc_1, HY_0 = tc_0 \cdot P$, and $HY_1 = tc_1 \cdot P$. Besides, $\mathcal{C}$ executes the above-mentioned hash function queries. Moreover,

Authorized licensed use limited to: NANJING NORMAL UNIVERSITY. Downloaded on December 07,2022 at 10:48:34 UTC from IEEE Xplore. Restrictions apply.

$\mathcal{C}$ calculates $US_{ID,t} = f_{-2} \cdot tc_0 + f_{-1} \cdot (tc_1 + x_{ID}) \bmod p$, where $t \equiv 1 \bmod 2$. Else, $\mathcal{C}$ calculates $US_{ID,t} = f_{-2} \cdot tc_1 + f_{-1} \cdot (tc_0 + x_{ID}) \bmod p$, where $t \equiv 0 \bmod 2$. Finally, $\mathcal{C}$ returns $US_{ID,t}$ to $\mathcal{A}_1$.

j) *Sign Query:* When receiving this query on ID_i, $\mathcal{C}$ first searches the tuple $(\mathrm{ID}_i, x_{ID}, UY_{ID}, x'_{ID}, UY'_{ID})$ from L_{pk} and does the following steps.
   i) If $\mathrm{ID}_i \neq ID^*$, $\mathcal{C}$ makes the works as follows.
      a) If this tuple does not exist, $\mathcal{C}$ executes the algorithm of *UserKeyExtract* to generate UY_{ID} and then inserts $(\mathrm{ID}_i, x_{ID}, UY_{ID}, -, -)$ into L_{pk}.
      b) If this tuple does exist, $\mathcal{C}$ delivers x_{ID} to $\mathcal{A}_1$.
      c) If this tuple does exist and the public key UY_{ID} has been replaced with UY'_{ID}, $\mathcal{C}$ sets x'_{ID} as the private key.

      Finally, $\mathcal{C}$ executes the *Sign* algorithm with $(Cert_i, US_{ID,t})$ as the input to produce σ_i as well as sends it to $\mathcal{A}_1$.
   ii) If $\mathrm{ID}_i = ID^*$, $\mathcal{C}$ first searches $(\mathrm{ID}_i, W_i, \Psi_i)$ from L_1 and $(\mathrm{ID}_i, m_i, \Delta, UY_{ID}, T_{pub}, W_i, K_i, \xi_i)$ from L_2. Then, $\mathcal{C}$ randomly picks $w_i, k_i \in Z_q^*$ and sets $K_i = k_iP - \xi_i^{-1}\Psi_i$, $W_i = w_iP - \xi_i^{-1}\Psi_i$, and $R_i = \xi_i \cdot k_i \cdot T_{pub} + \xi_i \cdot US_{ID,t} \cdot Q + w_iQ - \xi_i^{-1}tbP$. Finally, $\mathcal{C}$ returns (W_i, K_i, R_i) to $\mathcal{A}_1$.

3) *Forgery:* If $\mathrm{ID}_i \neq ID^*$, this algorithm aborts; otherwise, according to the forgery theorem, $\mathcal{A}_1$ can forgery two signatures $\sigma_1 = (W_i, K_i, R_i)$ and $\sigma_2 = (W_i, K_i, R'_i)$ and returns them to $\mathcal{C}$. Finally, $\mathcal{C}$ can obtain the following equations:

$$e(R_i, P) = e(\Psi_i + \xi_i \cdot K_i, T_{pub}) \cdot e(W_i + \xi_i \cdot \Theta_i, Q) \quad (1)$$
$$e(R'_i, P) = e(\Psi_i + \xi'_i \cdot K_i, T_{pub}) \cdot e(W_i + \xi'_i \cdot \Theta_i, Q). \quad (2)$$

Then, $\mathcal{C}$ can obtain the solution of CDHP by calculating $[(\xi_i \cdot R'_i - \xi'_i \cdot R_i - (\xi_i - \xi'_i)t \cdot W_i)/(\xi_i - \xi'_i)]$.

Theorem 2: The introduced CB-PKIAS is EUF-CMA under the attack, which launched by a type II adversary.

Proof: Given (P, aP, bP) as the input of CDHP, the target of $\mathcal{C}$ is to compute abP. This process is completed by the interaction between $\mathcal{A}_2$ and $\mathcal{C}$ as follows.

1) *Setup:* $\mathcal{C}$ first chooses d_1 and d_2 of length l and then it sets $Q = aP$. Finally, this algorithm generates the master public key $par = (l, d_1, d_2, H_0, H_1, H_2, H_3, H_4, G, G_T, P, Q, \Delta, T_{pub})$. $\mathcal{C}$ maintains the lists $L_0, L_1, L_2, L_3, L_4, L_{pk}$, and L_t.
2) *Query:* In this section, $\mathcal{A}_2$ does the following steps.
   a) *H0 Query:* When receiving this query on $d \in (d_1, d_2)$, $\mathcal{C}$ first checks if L_0 includes d. If it exists, $\mathcal{C}$ returns the stored outcome to $\mathcal{A}_1$. Otherwise, $\mathcal{C}$ picks $D_d \in G$, then it adds (d, D_d) into L_0 and returns D_d to $\mathcal{A}_2$.
   b) *H1 Query:* When receiving this query on (ID_i, W_i), $\mathcal{C}$ first checks if L_1 includes (ID_i, W_i). If it exists, $\mathcal{C}$ returns the stored outcome to $\mathcal{A}_2$. Otherwise, $\mathcal{C}$ picks $v \in Z_q^*$ and calculates $\Psi_i = vP$, then it adds $(\mathrm{ID}_i, W_i, \Psi_i)$ into L_1 and returns Ψ_i to $\mathcal{A}_2$.
   c) *H2 Query:* When receiving this query on $(\mathrm{ID}_i, m_i, \Delta, UY_{ID}, T_{pub}, W_i, K_i)$, $\mathcal{C}$ first checks if L_2 includes them. If it exists, $\mathcal{C}$ returns the stored outcome to $\mathcal{A}_2$. Otherwise, $\mathcal{C}$ picks $\xi_i \in Z_q^*$, then it adds $(\mathrm{ID}_i, m_i, \Delta, UY_{ID}, T_{pub}, W_i, K_i, \xi_i)$ into L_2 and returns ξ_i to $\mathcal{A}_2$.
   d) *H3 Query:* When receiving this query on (R_i, Y_{ver}), $\mathcal{C}$ first checks if L_3 includes them. If it exists, $\mathcal{C}$ returns the stored outcome to $\mathcal{A}_2$. Otherwise, $\mathcal{C}$ picks $j_i \in Z_q^*$, then it adds (R_i, Y_{ver}, j_i) into L_3 and returns j_i to $\mathcal{A}_2$.
   e) *H4 Query:* When receiving this query on $(\mathrm{ID}_i, UY_{ID}, -2)$ or $(\mathrm{ID}_i, UY_{ID}, -1)$, $\mathcal{C}$ first checks if L_4 includes them. If it exists, $\mathcal{C}$ returns the stored outcome to $\mathcal{A}_2$. Otherwise, $\mathcal{C}$ picks $f \in Z_q^*$ and adds it into L_4. Finally, $\mathcal{C}$ returns f to $\mathcal{A}_2$.
   f) *Public Key Query:* Suppose that L_{pk} stores the tuple structure $(\mathrm{ID}_i, x_{ID}, UY_{ID}, x'_{ID}, UY'_{ID})$. When receiving this query on ID_i, $\mathcal{C}$ does the following steps.
      i) If this tuple does not exist and $\mathrm{ID}_i = ID^*$, $\mathcal{C}$ sets $UY_{ID} = bP$. Finally, $\mathcal{C}$ inserts $(\mathrm{ID}_i, -, UY_{ID}, -, -)$ into L_{pk} and sends UY_{ID} to $\mathcal{A}_2$. Else, $\mathcal{C}$ picks $x_{ID} \in Z_q^*$ and returns $UY_{ID} = x_{ID}P$ to $\mathcal{A}_2$.
      ii) If this tuple does exist, $\mathcal{C}$ returns UY_{ID} to $\mathcal{A}_2$.
   g) *Helper Key Query:* When receiving this query on (ID_i, t), $\mathcal{C}$ maintains the list $L_t = \{\mathrm{ID}_i, HS_0, HS_1, HY_0, HY_1\}$. Afterward, $\mathcal{C}$ checks if ID_i exists in L_t. If it does not exist, $\mathcal{C}$ randomly picks $tc_0, tc_1 \in Z_q^*$, then calculates $HS_0 = tc_0, HS_1 = tc_1, HY_0 = tc_0 \cdot P, HY_1 = tc_1 \cdot P$. Besides, $\mathcal{C}$ executes the above-mentioned hash function queries. Finally, $\mathcal{C}$ returns HS_0, HS_1, HY_0, HY_1 to $\mathcal{A}_2$.
   h) *Sign Query:* When receiving this query on ID_i, $\mathcal{C}$ first searches the tuple $(\mathrm{ID}_i, x_{ID}, UY_{ID}, -, -)$ for x_{ID} from L_{pk} and does the following steps.
      i) If $x_{ID} \neq \perp$, that is to say, $\mathcal{C}$ can run the *Sign* algorithm with $US_{ID,t}$ and $Cert_i$.
      ii) If $x_{ID} = \perp$, $\mathcal{C}$ first searches $(\mathrm{ID}_i, W_i, \Psi_i)$ from L_1 and $(\mathrm{ID}_i, m_i, \Delta, UY_{ID}, T_{pub}, W_i, K_i, \xi_i)$ from L_2. Then, $\mathcal{C}$ randomly picks $k_i, t \in Z_q^*$. Then, $\mathcal{C}$ calculates $K_i = k_iP$, $W_i = tT_{pub} - \xi_i\Theta_i$, and $R_i = s\Psi_i + \xi_i \cdot s \cdot K_i + t \cdot s \cdot Q$ and returns (W_i, K_i, R_i) to $\mathcal{A}_2$.

3) *Forgery:* If $\mathrm{ID}_i \neq ID^*$, this algorithm aborts; otherwise, according to the forgery theorem, $\mathcal{A}_2$ can forgery two signatures $\sigma_1 = (W_i, K_i, R_i)$ and $\sigma_2 = (W_i, K_i, R'_i)$ and returns them to $\mathcal{C}$. Finally, $\mathcal{C}$ can obtain the following equations:

$$e(R_i, P) = e(\Psi_i + \xi_i \cdot K_i, T_{pub}) \cdot e(W_i + \xi_i \cdot \Theta_i, Q) \quad (3)$$
$$e(R'_i, P) = e(\Psi_i + \xi'_i \cdot K_i, T_{pub}) \cdot e(W_i + \xi'_i \cdot \Theta_i, Q). \quad (4)$$

Then, $\mathcal{C}$ can obtain the solution of CDHP by calculating $[(\xi_i \cdot R'_i - \xi'_i \cdot R_i - (\xi_i - \xi'_i)t \cdot W_i)/(\xi_i - \xi'_i)]$.

Authorized licensed use limited to: NANJING NORMAL UNIVERSITY. Downloaded on December 07,2022 at 10:48:34 UTC from IEEE Xplore. Restrictions apply.

Theorem 3: The proposed CB-PKIAS scheme can resist the fully chosen-key attacks, which launched by a fully chosen-key attacker $\mathcal{A}_3$.

Proof: If a fully chosen-key attacker $\mathcal{A}_3$ with the advantage of ϵ can break the validation, that is to say, there is a challenger $\mathcal{C}$ that has the ability to destroy the collision resistance. The concrete interactions are demonstrated as follows.

1) *Setup:* $\mathcal{C}$ executes the algorithm of *Setup* to produce s and $par = (l, d_1, d_2, H_0, H_1, H_2, H_3, H_4, G, G_T, P, Q, \Delta, T_{pub})$. Besides, it randomly picks τ and calculates $Y_{ver} = \tau P$ as the corresponding private verification key and public verification key, respectively.
2) *Query:* $\mathcal{A}_3$ mainly makes the following steps.
   a) *Signing Key Query:* When submitting the query on ID_i, $\mathcal{C}$ executes the algorithms *UserKeyExtract*, *Update**, and *Update*. Finally, $\mathcal{C}$ transfers $US_{ID,t}$ to $\mathcal{A}_3$.
   b) *AggVer Query:* When submitting this query, $\mathcal{C}$ executes the algorithm *AggVer* and delivers the verification result to $\mathcal{A}_3$.
3) *Forgery:* In this section, $\mathcal{A}_3$ can forge the aggregate signature $\{(\mathrm{ID}_i, UY_{ID}, m_i, \sigma_i)|i = 1, \ldots, n\}$. Moreover, it can resist the fully chosen-key attacks by satisfying the conditions as follows.
   a) All single signatures are aggregated into the aggregate signatures Σ. In this condition, $\gamma = H_3(e(R_1, Y_{ver})|| \cdots ||e(R_n, Y_{ver}))$.
   b) The aforementioned Σ is valid. In this condition, $\gamma = H_3\begin{pmatrix} e(\Psi_1 + \xi_1 \cdot K_1, \tau \cdot T_{pub}) \cdot e(W_1 + \xi_1 \cdot \Theta_1, \tau \cdot Q)|| \cdots \\ ||e(\Psi_n + \xi_n \cdot K_n, \tau \cdot T_{pub}) \cdot e(W_n + \xi_n \cdot \Theta_n, \tau \cdot Q) \end{pmatrix}$.
   c) At least one signature σ_i' cannot hold the verification. In particular, $e(R_i', P) \neq e(\Psi_i' + \xi_i' \cdot K_i', T_{pub}) \cdot e(W_i' + \xi_i' \cdot \Theta_i', Q)$ Thus, we can obtain $e(R_i', \tau P) \neq e(\Psi_i' + \xi_i' \cdot K_i', \tau T_{pub}) \cdot e(W_i' + \xi_i' \cdot \Theta_i', \tau Q)$. It is evident that the γ value from $\gamma = H_3(e(R_1, Y_{ver})|| \cdots ||e(R_n, Y_{ver}))$ is the same as $\gamma = H_3\begin{pmatrix} e(\Psi_1 + \xi_1 \cdot K_1, \tau \cdot T_{pub}) \cdot e(W_1 + \xi_1 \cdot \Theta_1, \tau \cdot Q)|| \cdots \\ ||e(\Psi_n + \xi_n \cdot K_n, \tau \cdot T_{pub}) \cdot e(W_n + \xi_n \cdot \Theta_n, \tau \cdot Q) \end{pmatrix}$, which is a contradictory to the fact that a single signature cannot pass the verification.

Theorem 4: The introduced CB-PKIAS is EUF-CMA and strongly key-insulated secure under this attack.

Proof: Given (P, aP, bP) as the input of CDHP, the target of $\mathcal{C}$ is to compute abP. This process is completed by the interaction between $\mathcal{A}_4$ and $\mathcal{C}$ as follows.

1) *Setup:* $\mathcal{C}$ chooses d_1 and d_2 of length l, then it selects $t \in Z_q^*$ and calculates $Q = tP$. Besides, $\mathcal{C}$ sets $T_{pub} = aP$. Finally, this algorithm generates $par = (l, d_1, d_2, H_0, H_1, H_2, H_3, H_4, G, G_T, P, Q, \Delta, T_{pub})$. $\mathcal{C}$ maintains the lists $L_0, L_1, L_2, L_3, L_4, L_{pk}$, and L_t.
2) *Query:* In this section, $\mathcal{A}_4$ performs a similar operation as shown in $\mathcal{A}_1$, except for the following query.
   a) *Helper Key Query:* When receiving this query on (ID_i, t), $\mathcal{C}$ maintains the list $L_t = \{\mathrm{ID}_i, HS_0, HS_1, HY_0, HY_1\}$. Afterward, $\mathcal{C}$ checks if ID_i exists in L_t. If so, $\mathcal{C}$ randomly chooses $tc_0, tc_1 \in Z_q^*$, then calculates $HS_0 = tc_0, HS_1 = tc_1, HY_0 = tc_0 \cdot P$, and $HY_1 = tc_1 \cdot P$. Besides, $\mathcal{C}$ executes the abovementioned hash function queries. Finally, $\mathcal{C}$ returns HS_0, HS_1, HY_0, and HY_1 to $\mathcal{A}_4$.
3) *Forgery:* If $\mathrm{ID}_i \neq ID^*$, this algorithm aborts; otherwise, according to the forgery theorem, $\mathcal{A}_4$ can forgery two signatures $\sigma_1 = (W_i, K_i, R_i)$ and $\sigma_2 = (W_i, K_i, R_i')$ and returns them to $\mathcal{C}$. Finally, $\mathcal{C}$ can obtain the following equations:

$$e(R_i, P) = e(\Psi_i + \xi_i \cdot K_i, T_{pub}) \cdot e(W_i + \xi_i \cdot \Theta_i, Q) \quad (5)$$
$$e(R_i', P) = e(\Psi_i + \xi_i' \cdot K_i, T_{pub}) \cdot e(W_i + \xi_i' \cdot \Theta_i, Q). \quad (6)$$

Then, $\mathcal{C}$ can obtain the solution of CDHP by calculating $[(\xi_i \cdot R_i' - \xi_i' \cdot R_i - (\xi_i - \xi_i')t \cdot W_i)/(\xi_i - \xi_i')]$.

Theorem 5: The introduced CB-PKIAS is EUF-CMA and strongly key-insulated secure under the attack.

Proof: Given (P, aP, bP) as the input of CDHP, the target of $\mathcal{C}$ is to compute abP. This process is completed by the interaction between $\mathcal{A}_5$ and $\mathcal{C}$ as follows.

1) *Setup:* $\mathcal{C}$ chooses d_1 and d_2 of length l, then it sets $Q = aP$. Finally, this algorithm generates $par = (l, d_1, d_2, H_0, H_1, H_2, H_3, H_4, G, G_T, P, Q, \Delta, T_{pub})$. $\mathcal{C}$ maintains the lists $L_0, L_1, L_2, L_3, L_4, L_{pk}$, and L_t.
2) *Query:* In this section, $\mathcal{A}_5$ performs a similar operation as shown in $\mathcal{A}_2$, except for the following query.
   a) *Helper Key Query:* When receiving this query on (ID_i, t), $\mathcal{C}$ maintains the list $L_t = \{\mathrm{ID}_i, HS_0, HS_1, HY_0, HY_1\}$. Afterward, $\mathcal{C}$ checks if ID_i exists in L_t. If so, $\mathcal{C}$ randomly chooses $tc_0, tc_1 \in Z_q^*$, then calculates $HS_0 = tc_0, HS_1 = tc_1, HY_0 = tc_0 \cdot P$, and $HY_1 = tc_1 \cdot P$. Besides, $\mathcal{C}$ executes the abovementioned hash function queries. Finally, $\mathcal{C}$ returns HS_0, HS_1, HY_0, and HY_1 to $\mathcal{A}_2$.
3) *Forgery:* If $\mathrm{ID}_i \neq ID^*$, this algorithm aborts; otherwise, according to the forgery theorem, $\mathcal{A}_5$ can forgery two signatures $\sigma_1 = (W_i, K_i, R_i)$ and $\sigma_2 = (W_i, K_i, R_i')$ and returns them to $\mathcal{C}$. Finally, $\mathcal{C}$ can obtain the following equations:

$$e(R_i, P) = e(\Psi_i + \xi_i \cdot K_i, T_{pub}) \cdot e(W_i + \xi_i \cdot \Theta_i, Q) \quad (7)$$
$$e(R_i', P) = e(\Psi_i + \xi_i' \cdot K_i, T_{pub}) \cdot e(W_i + \xi_i' \cdot \Theta_i, Q). \quad (8)$$

Then, $\mathcal{C}$ can obtain the solution of CDHP by calculating $[(\xi_i \cdot R_i' - \xi_i' \cdot R_i - (\xi_i - \xi_i')t \cdot W_i)/(\xi_i - \xi_i')]$.

VII. Comparison

In this section, we demonstrated the performance of our proposed scheme with the competitive works in [13], [14], [36], [37], [44], and [50]–[53]. As Table IV shows, we give the concrete comparison in terms of the key insulated, aggregation, security level, security assumption, and whether resist the fully chosen-key attacks. Besides, it is important to note that the symbol "×" refers to that the corresponding scheme cannot achieve this property and the symbol "✓" represents that the corresponding scheme has this ability. Obviously, our scheme can achieve all the properties.

For describing the computation efficiency accurately, we employ the computer with Intel Core i5-8400 CPU @

Authorized licensed use limited to: NANJING NORMAL UNIVERSITY. Downloaded on December 07,2022 at 10:48:34 UTC from IEEE Xplore. Restrictions apply.

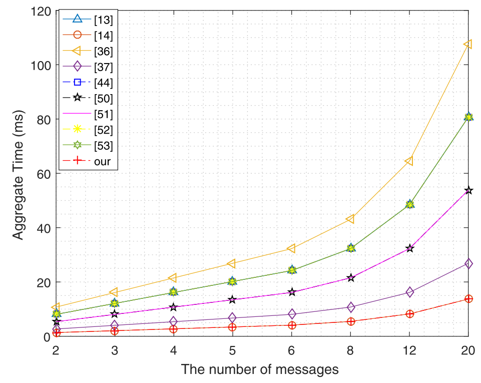


Fig. 3. Aggregation consumption.

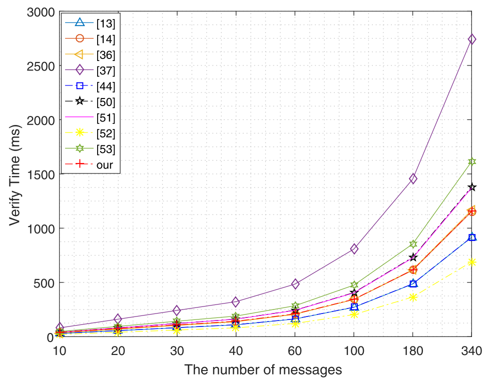


Fig. 4. Verification consumption.

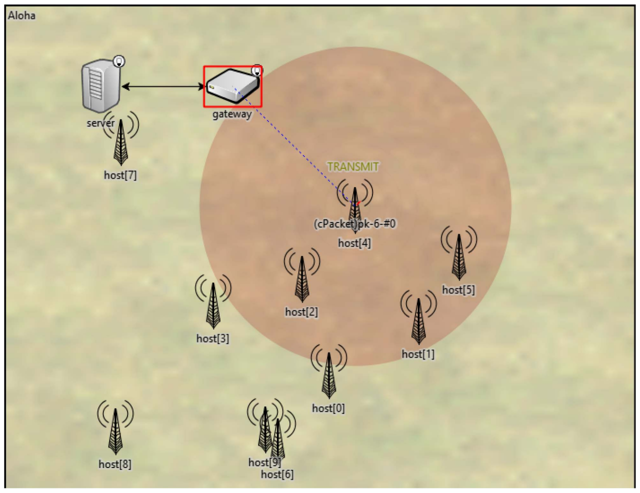


Fig. 5. Concrete network topology.

2.80 GHz as well as 16.00 GB to execute the simulation experiment. Then, we integrated the PBC library into the VMware Workstation Pro 14. Besides, in order to achieve the 1024-b RSA, a curve $y^2 = x^3 + x$ is utilized, where the embedding degree is 2, $q = 2^{159} + 2^{17} + 1$ is considered as a prime, and $p = 12qr - 1$ refers to the 512-b prime. Therefore, after the repeated simulation experiments, the concrete running time is demonstrated in Table II. In addition, we can acquire $|Z_q^*| =$ 20 B and $|G| = |G_T| =$128 B. Furthermore, Table I is shown to explain the meanings corresponding to the specific symbol.

TABLE I
SYMBOLS AND CORRESPONDING DESCRIPTIONS

Notation	Description
τ	The secret verification key
Y_{ver}	The public verification key
TA	Trusted authority
s	The master secret key
G, G_T	Two cyclic additive group
q	Prime order
H_0,H_1,H_2,H_3,H_4	Hash function
d_1, d_2, l	Two bit strings and the length
T_{pub}	The system public key
Δ	The state information
par	Parameter
ID_i	The identity of user i
$Cert_i$	The certificate of user i
t	Time period
$US_{ID,0}$, UY_{ID}	The initial secret/public key
(HS_0, HY_0)	The secret/public key of helper 0
(HS_1, HY_1)	The secret/public key of helper 1
x_{ID}	The secret value
$UD_{ID,t}$	The update key
$US_{ID,t}$	The temporary signing key
σ_i	The single signature
Σ	The aggregate signature
T_p	The pairing operation
T_a	The point additive operation in G
T_e	The exponentiation operation in G_T
T_h	The operation of hash function

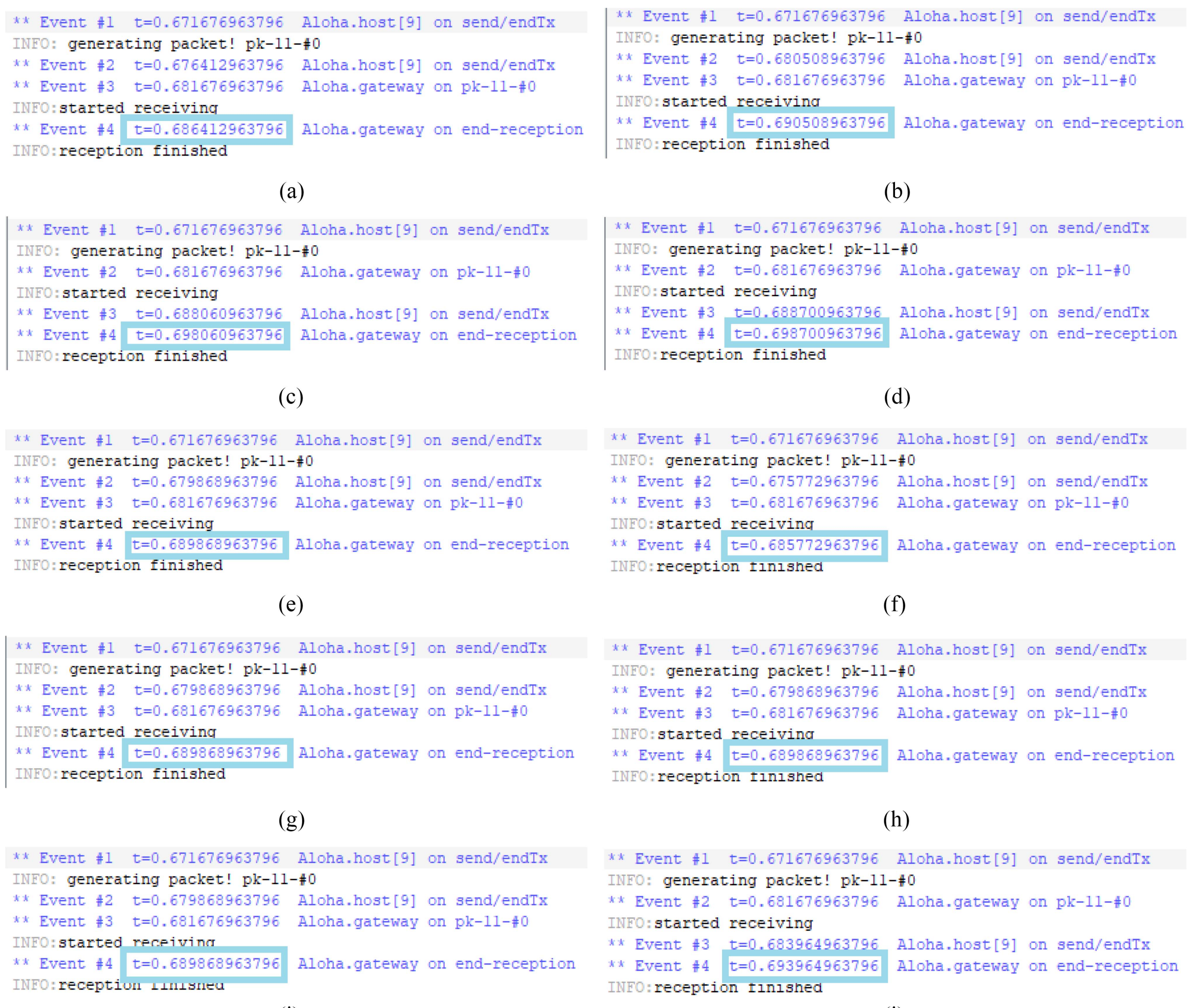
TABLE II
EXECUTION TIMES OF CRYPTOGRAPHIC OPERATIONS

Operation	T_p	T_a	T_e
Time(ms)	0.6858	1.3449	0.0949

Furthermore, we use the OMNeT++ event simulator to simulate the signature transmission of our scheme [54]. The simulation is based on the aloha protocol. Then, the link communication rate, transmission delay and the network scale were set to 250 kb/s, 10ms, 3000–15 000, respectively. It is worth mentioning that the ZigBee's default data rate is 250 kb/s, which is a general setting toward IIoT devices. In addition, Fig. 5 demonstrates our simulation experiment's network topology. In this figure, three entities, such as nodes, gateways, and servers are included to describe the workflow. For example, the node is the sensor node, which is responsible for collecting data in IIoT and sending the data to the gateway through the wireless network. If the data are received by the gateway, the forward function will start and the data will be delivered to the server. Therefore, the transmission time is an obvious indicator for evaluating the communication overhead. Subsequently, the message will be stored and processed on the server.

The communication overheads of different schemes are described in Table III and Fig. 6. Obviously, CB-PKIAS has a shorter private key than [14], [36], [44], [50]–[53]. Besides, the size of aggregate signature in our scheme is smaller than competitive schemes [36], [37] and slightly longer than the other schemes, which is normal since the CB-PKIAS adds the functionality of the fully chosen attacks on this basis. Furthermore, we also simulate the overhead of signature transmission, which

Authorized licensed use limited to: NANJING NORMAL UNIVERSITY. Downloaded on December 07,2022 at 10:48:34 UTC from IEEE Xplore. Restrictions apply.



```
** Event #1  t=0.671676963796  Aloha.host[9] on send/endTx
INFO: generating packet! pk-11-#0
** Event #2  t=0.676412963796  Aloha.host[9] on send/endTx
** Event #3  t=0.681676963796  Aloha.gateway on pk-11-#0
INFO:started receiving
** Event #4  t=0.686412963796  Aloha.gateway on end-reception
INFO:reception finished
```

(a)

```
** Event #1  t=0.671676963796  Aloha.host[9] on send/endTx
INFO: generating packet! pk-11-#0
** Event #2  t=0.680508963796  Aloha.host[9] on send/endTx
** Event #3  t=0.681676963796  Aloha.gateway on pk-11-#0
INFO:started receiving
** Event #4  t=0.690508963796  Aloha.gateway on end-reception
INFO:reception finished
```

(b)

```
** Event #1  t=0.671676963796  Aloha.host[9] on send/endTx
INFO: generating packet! pk-11-#0
** Event #2  t=0.681676963796  Aloha.gateway on pk-11-#0
INFO:started receiving
** Event #3  t=0.688060963796  Aloha.host[9] on send/endTx
** Event #4  t=0.698060963796  Aloha.gateway on end-reception
INFO:reception finished
```

(c)

```
** Event #1  t=0.671676963796  Aloha.host[9] on send/endTx
INFO: generating packet! pk-11-#0
** Event #2  t=0.681676963796  Aloha.gateway on pk-11-#0
INFO:started receiving
** Event #3  t=0.688700963796  Aloha.host[9] on send/endTx
** Event #4  t=0.698700963796  Aloha.gateway on end-reception
INFO:reception finished
```

(d)

```
** Event #1  t=0.671676963796  Aloha.host[9] on send/endTx
INFO: generating packet! pk-11-#0
** Event #2  t=0.679868963796  Aloha.host[9] on send/endTx
** Event #3  t=0.681676963796  Aloha.gateway on pk-11-#0
INFO:started receiving
** Event #4  t=0.689868963796  Aloha.gateway on end-reception
INFO:reception finished
```

(e)

```
** Event #1  t=0.671676963796  Aloha.host[9] on send/endTx
INFO: generating packet! pk-11-#0
** Event #2  t=0.675772963796  Aloha.host[9] on send/endTx
** Event #3  t=0.681676963796  Aloha.gateway on pk-11-#0
INFO:started receiving
** Event #4  t=0.685772963796  Aloha.gateway on end-reception
INFO:reception finished
```

(f)

```
** Event #1  t=0.671676963796  Aloha.host[9] on send/endTx
INFO: generating packet! pk-11-#0
** Event #2  t=0.679868963796  Aloha.host[9] on send/endTx
** Event #3  t=0.681676963796  Aloha.gateway on pk-11-#0
INFO:started receiving
** Event #4  t=0.689868963796  Aloha.gateway on end-reception
INFO:reception finished
```

(g)

```
** Event #1  t=0.671676963796  Aloha.host[9] on send/endTx
INFO: generating packet! pk-11-#0
** Event #2  t=0.679868963796  Aloha.host[9] on send/endTx
** Event #3  t=0.681676963796  Aloha.gateway on pk-11-#0
INFO:started receiving
** Event #4  t=0.689868963796  Aloha.gateway on end-reception
INFO:reception finished
```

(h)

```
** Event #1  t=0.671676963796  Aloha.host[9] on send/endTx
INFO: generating packet! pk-11-#0
** Event #2  t=0.679868963796  Aloha.host[9] on send/endTx
** Event #3  t=0.681676963796  Aloha.gateway on pk-11-#0
INFO:started receiving
** Event #4  t=0.689868963796  Aloha.gateway on end-reception
INFO:reception finished
```

(i)

```
** Event #1  t=0.671676963796  Aloha.host[9] on send/endTx
INFO: generating packet! pk-11-#0
** Event #2  t=0.681676963796  Aloha.gateway on pk-11-#0
INFO:started receiving
** Event #3  t=0.683964963796  Aloha.host[9] on send/endTx
** Event #4  t=0.693964963796  Aloha.gateway on end-reception
INFO:reception finished
```

(j)

Fig. 6. Signature transmission overhead simulation compared with related schemes. (a) Signature overhead of [13] in transmission process. (b) Signature overhead of [14] in transmission process. (c) Signature overhead of [36] in transmission process. (d) Signature overhead of [37] in transmission process. (e) Signature overhead of [44] in transmission process. (f) Signature overhead of [50] in transmission process. (g) Signature overhead of [51] in transmission process. (h) Signature overhead of [52] in transmission process. (i) Signature overhead of [53] in transmission process. (j) Signature overhead of our scheme in transmission process.

TABLE III
COMPARISON THE OVERHEADS OF COMMUNICATION AND COMPUTATION

Scheme	Aggregate	Verify	Private key	Public key	Aggregate signature
[13]	3nT_a	(2n+1)T_a	$\lvert Z_q^*\rvert$	$\lvert G\rvert$	$\lvert G\rvert$+$\lvert Z_q^*\rvert$
[14]	nT_p	(n+3)T_p+(2n+2)T_a	$\lvert G\rvert$+$\lvert Z_q^*\rvert$	$\lvert G\rvert$	(n+1)$\lvert G\rvert$+$\lvert Z_q^*\rvert$
[36]	4nT_a	5nT_p	3$\lvert G\rvert$	2$\lvert G\rvert$	4n$\lvert G\rvert$
[37]	nT_a	6nT_a	$\lvert Z_q^*\rvert$	$\lvert G\rvert$	4n$\lvert G\rvert$+n$\lvert Z_q^*\rvert$
[44]	3nT_a	4T_p+2nT_a	$\lvert G\rvert$+$\lvert Z_q^*\rvert$	$\lvert G\rvert$	2$\lvert G\rvert$
[50]	2nT_a	2nT_p+2nT_a	2$\lvert Z_q^*\rvert$	2$\lvert G\rvert$	n$\lvert G\rvert$
[51]	2nT_a	3nT_a	2$\lvert Z_q^*\rvert$	2$\lvert G\rvert$	2$\lvert G\rvert$
[52]	3nT_a	nT_p+nT_a	$\lvert G\rvert$+2$\lvert Z_q^*\rvert$	2$\lvert G\rvert$	2n$\lvert G\rvert$
[53]	3nT_a	3nT_p+2nT_a	$\lvert G\rvert$	$\lvert G\rvert$	2$\lvert G\rvert$
Our	nT_p	(n+3)T_p+(2n+4)T_a	$\lvert Z_q^*\rvert$	$\lvert G\rvert$	(2n+1)$\lvert G\rvert$

* Legends: $\lvert G\rvert$: a point's size in G, $\lvert Z_q^*\rvert$: a bit length in Z_q^*.

is consistent with the above analysis results. Therefore, our proposed scheme is secure and preferable toward IIoT.

To evaluate the efficiency in our scheme and existing works, Table III and Figs. 3 and 4 are demonstrated, which calculate the concrete value of the computational overhead. Specifically, the Fig. 3 shows the concrete time overhead in terms of the aggregate algorithm. In this operation, n signatures are aggregated into an aggregate signature. Meanwhile, Fig. 4 describes

Authorized licensed use limited to: NANJING NORMAL UNIVERSITY. Downloaded on December 07,2022 at 10:48:34 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
COMPARISON THE PERFORMANCES OF DIFFERENT SCHEMES

Scheme	Security against $\mathcal{A}_1$	Security against $\mathcal{A}_2$	Resist FCA	Key-insulated	Aggregation	Hard problem
[13]	×	×	×	×	✓	ECDLP
[14]	✓	✓	✓	×	✓	CDHP
[36]	✓	✓	×	✓	×	Many-DHP & NGBDHP
[37]	✓	✓	×	✓	×	DLP
[44]	✓	✓	×	×	✓	CDHP
[50]	✓	✓	×	×	×	q-CAAP
[51]	✓	✓	×	×	✓	ECDLP&CDHP
[52]	✓	✓	×	×	×	CDHP&BSDHP&EBSDHP
[53]	✓	✓	×	✓	✓	CDHP&DDHP&GDHP
Our	✓	✓	✓	✓	✓	CDHP

* Legends: FCA: fully chosen-key attacks, ECDLP: elliptic curve discrete log problem, DLP: discrete logarithm problem, Many-DHP: many diffie-hellman problem, NGBDHP: non pairing-based generalized bilinear DH problem, CDHP: computational Diffie-Hellman problem.

the time overhead of verification, which gives the contrast effect directly. It is clear that the aggregation consumption in CB-PKIAS is much lower than the schemes in [13], [36], [37], [44], and [50]–[53] and the same as the scheme in [14]. Besides, the overhead of verification assumption in CB-PKIAS is slightly expensive, which is tolerant since the proposed scheme extends the functions of key insulated and secure against the fully chosen attacks that the comparative scheme does not have.

VIII. Conclusion

This article analyzed the security of Verma *et al.*'s protocol toward the IIoT environment and illustrated that the attacker from a malicious KGC, the public-key replacement, the fully chosen, as well as the outsider all can generate a forged signature without being detected. Therefore, we demonstrated that the scheme in [13] fails to achieve the claimed security features. Afterward, the corresponding guideline was given to guarantee the certificate-based signature's security. Finally, we suggested CB-PKIAS for IIoT, which not only preserves the unforgeability of transmitted messages but also resists the fully chosen attack and avoids the key exposure. The rigorous simulation and careful analysis demonstrated that the introduced scheme is more suitable for the IIoT environment.

References

[1] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain, "Machine learning-based network vulnerability analysis of industrial Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6822–6834, Aug. 2019.

[2] F. Al-Turjman and S. Alturjman, "Context-sensitive access in industrial Internet of Things (IIoT) healthcare applications," *IEEE Trans. Ind. Informat.*, vol. 14, no. 6, pp. 2736–2744, Jun. 2018.

[3] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proc. Workshop Theory Appl. Cryptograph. Techn.*, 1984, pp. 47–53.

[4] F. Hess, "Efficient identity based signature schemes based on pairings," in *Proc. Int. Workshop Sel. Areas Cryptogr.*, 2002, pp. 310–324.

[5] J. C. Choon and J. H. Cheon, "An identity-based signature from gap Diffie-Hellman groups," in *Proc. Int. Workshop Public Key Cryptogr.*, 2003, pp. 18–30.

[6] X. Yi, "An identity-based signature scheme from the Weil pairing," *IEEE Commun. Lett.*, vol. 7, no. 2, pp. 76–78, Feb. 2003.

[7] A. M. Fischer, "Public key/signature cryptosystem with enhanced digital signature certification," U.S. Patent CA2 000 400C, 1989.

[8] I. Ali, M. Gervais, E. Ahene, and F. Li, "A blockchain-based certificateless public key signature scheme for vehicle-to-infrastructure communication in VANETs," *J. Syst. Archit.*, vol. 99, Oct. 2019, Art. no. 101636.

[9] C. Xie, J. Weng, J. Weng, and L. Hou, "Scalable revocable identity-based signature over lattices in the standard model," *Inf. Sci.*, vol. 518, pp. 29–38, May 2020.

[10] B. G. Kang, J. H. Park, and S. G. Hahn, *A Certificate-Based Signature Scheme*, vol. 2964. Heidelberg, Germany: Springer, 2004, pp. 99–111.

[11] G. K. Verma, B. B. Singh, N. Kumar, M. S. Obaidat, D. He, and H. Singh, "An efficient and provable certificate-based proxy signature scheme for IIoT environment," *Inf. Sci.*, vol. 518, pp. 142–156, May 2020.

[12] J. K. Liu, J. Baek, and J. Zhou, "Certificate-based sequential aggregate signature," in *Proc. 2nd ACM Conf. Wireless Netw. Security*, 2009, pp. 21–28.

[13] G. K. Verma, B. B. Singh, N. Kumar, and V. Chamola, "CB-CAS: Certificate-based efficient signature scheme with compact aggregation for industrial Internet of Things environment," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2563–2572, Apr. 2020.

[14] G. Wu, F. Zhang, L. Shen, F. Guo, and W. Susilo, "Certificateless aggregate signature scheme secure against fully chosen-key attacks," *Inf. Sci.*, vol. 514, pp. 288–301, Apr. 2020.

[15] Y. Dodis, J. Katz, S. Xu, and M. Yung, "Key-insulated public key cryptosystems," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2002, pp. 65–82.

[16] G. Hanaoka, Y. Hanaoka, and H. Imai, "Parallel key-insulated public key encryption," in *Proc. Int. Workshop Public Key Cryptogr.*, 2006, pp. 105–122.

[17] I. Ali, T. Lawrence, and F. Li, "An efficient identity-based signature scheme without bilinear pairing for vehicle-to-vehicle communication in VANETs," *J. Syst. Archit.*, vol. 103, Feb. 2020, Art. no. 101692. doi: 10.1016/j.sysarc.2019.101692.

[18] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory*, vol. 22, no. 6, pp. 644–654, Nov. 1976.

[19] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[20] T. E. Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inf. Theory*, vol. 31, no. 4, pp. 469–472, Jul. 1985.

[21] V. S. Miller, "Use of elliptic curves in cryptography," in *Proc. Conf. Theory Appl. Cryptograph. Techn.*, 1985, pp. 417–426.

[22] A. Shamir, *Identity-Based Cryptosystems and Signature Schemes*, vol. 196. Heidelberg, Germany: Springer, 1984, pp. 47–53.

[23] R. Yaduvanshi and S. Mishra, "An efficient and secure pairing free short id-based signature scheme over elliptic curve," *SSRN Electron. J.*, 2019. doi: 10.2139/ssrn.3351027.

[24] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," in *Proc. 21st Annu. Int. Cryptol. Conf. Adv. Cryptol.*, 2001, pp. 213–229.

[25] H. Singh and G. K. Verma, "ID-based proxy signature scheme with message recovery," *J. Syst. Softw.*, vol. 85, no. 1, pp. 209–214, 2012.

[26] R. Tso, C. Gu, T. Okamoto, and E. Okamoto, *Efficient ID-Based Digital Signatures With Message Recovery*, vol. 4856. Heidelberg, Germany: Springer, 2007, pp. 47–59.

Authorized licensed use limited to: NANJING NORMAL UNIVERSITY. Downloaded on December 07,2022 at 10:48:34 UTC from IEEE Xplore. Restrictions apply.

[27] H. Du and Q. Wen, "An efficient identity-based short signature scheme from bilinear pairings," in *Proc. Int. Conf. Comput. Intell. Security (CIS)*, 2007, pp. 725–729.
[28] J. K. Liu, J. Baek, J. Zhou, Y. Yang, and J. W. Wong, "Efficient online/offline identity-based signature for wireless sensor network," *Int. J. Inf. Security*, vol. 9, no. 4, pp. 287–296, 2010.
[29] C. Gentry, *Certificate-Based Encryption and the Certificate Revocation Problem*, vol. 2656. Heidelberg, Germany: Springer, 2003, pp. 272–293.
[30] J. Li, X. Huang, Y. Mu, W. Susilo, and Q. Wu, *Certificate-Based Signature: Security Model and Efficient Construction*, vol. 4582. Heidelberg, Germany: Springer, 2007, pp. 110–125.
[31] J. Li, X. Huang, Y. Zhang, and L. Xu, "An efficient short certificate-based signature scheme," *J. Syst. Softw.*, vol. 85, no. 2, pp. 314–322, 2012.
[32] J. Li, Z. Wang, and Y. Zhang, "Provably secure certificate-based signature scheme without pairings," *Inf. Sci.*, vol. 233, pp. 313–320, Jun. 2013.
[33] C. Zhou and Z. Cui, "Certificate-based signature scheme in the standard model," *IET Inf. Security*, vol. 11, no. 5, pp. 256–260, Sep. 2017.
[34] H. Du, J. Li, Y. Zhang, T. Li, and Y. Zhang, "Certificate-based key-insulated signature," in *Proc. Int. Conf. Data Knowl. Eng.*, 2012, pp. 206–220.
[35] H. Xiong, S. Wu, J. Geng, E. Ahene, S. Wu, and Z. Qin, "A pairing-free key-insulated certificate-based signature scheme with provable security," *KSII Trans. Internet Inf. Syst.*, vol. 9, no. 3, pp. 1246–1259, 2015.
[36] J. Li, H. Du, and Y. Zhang, "Certificate-based key-insulated signature in the standard model," *Comput. J.*, vol. 59, no. 7, pp. 1028–1039, 2016.
[37] H. Xiong, Q. Mei, and Y. Zhao, "Efficient and provably secure certificateless parallel key-insulated signature without pairing for IIoT environments," *IEEE Syst. J.*, vol. 14, no. 1, pp. 310–320, Mar. 2020.
[38] Y. Watanabe and J. Shikata, "Identity-based hierarchical key-insulated encryption without random oracles," in *Public-Key Cryptography (PKC)*. Heidelberg, Germany: Springer, 2016, pp. 255–279.
[39] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, *Aggregate and Verifiably Encrypted Signatures From Bilinear Maps*, vol. 2656. Heidelberg, Germany: Springer, 2003, pp. 416–432.
[40] X. Ma, J. Shao, C. Zuo, and R. Meng, *Efficient Certificate-Based Signature and Its Aggregation*, vol. 10701. Cham, Switzerland: Springer, 2017, pp. 391–408.
[41] G. K. Verma, B. B. Singh, N. Kumar, O. Kaiwartya, and M. S. Obaidat, "PFCBAS: Pairing free and provable certificate-based aggregate signature scheme for the e-healthcare monitoring system," *IEEE Syst. J.*, vol. 14, no. 2, pp. 1704–1715, Jun. 2020.
[42] J. Li, Y. Zhang, J. Ning, X. Huang, G. S. Poh, and D. Wang, "Attribute based encryption with privacy protection and accountability for CloudIoT," *IEEE Trans. Cloud Comput.*, early access, Feb. 19, 2020, doi: 10.1109/TCC.2020.2975184.
[43] S. Belguith, N. Kaaniche, M. Hammoudeh, and T. Dargahi, "PROUD: Verifiable privacy-preserving outsourced attribute based signcryption supporting access policy update for cloud assisted IoT applications," *Future Gener. Comput. Syst.*, vol. 111, pp. 899–918, Oct. 2020.
[44] Q. Mei, H. Xiong, J. Chen, M. Yang, S. Kumari, and M. K. Khan, "Efficient certificateless aggregate signature with conditional privacy preservation in IoV," *IEEE Syst. J.*, early access, Feb. 25, 2020, doi: 10.1109/JSYST.2020.2966526.
[45] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2001, pp. 453–474.
[46] M. Wazid, P. Bagga, A. K. Das, S. Shetty, J. J. P. C. Rodrigues, and Y. H. Park, "AKM-IoV: Authenticated key management protocol in fog computing-based Internet of vehicles deployment," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8804–8817, Oct. 2019.
[47] D. Abbasinezhad-Mood, A. Ostad-Sharif, and M. Nikooghadam, "Novel anonymous key establishment protocol for isolated smart meters," *IEEE Trans. Ind. Electron.*, vol. 67, no. 4, pp. 2844–2851, Apr. 2020.
[48] K.-A. Shim, "Forgery attacks on two provably secure certificateless signature schemes," *Inf. Sci.*, vol. 521, pp. 81–87, Jun. 2020.
[49] W. Yang, S. Wang, and Y. Mu, "An enhanced certificateless aggregate signature without pairings for E-Healthcare system," *IEEE Internet Things J.*, early access, Oct. 28, 2020, doi: 10.1109/JIOT.2020.3034307.
[50] Y. Lu, G. Wang, J. Li, and J. Shen, "New construction of short certificate-based signature against existential forgery attacks," *KSII Trans. Internet Inf. Syst.*, vol. 11, no. 7, pp. 3629–3647, 2017.
[51] J. Liu, L. Wang, and Y. Yu, "Improved security of a pairing-free certificateless aggregate signature in healthcare wireless medical sensor networks," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5256–5266, Jun. 2020.
[52] A. Karati, S. H. Islam, and M. Karuppiah, "Provably secure and lightweight certificateless signature scheme for IIoT environments," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3701–3711, Aug. 2018.
[53] P. V. Reddy and P. V. S. S. N. Gopal, "Identity-based key-insulated aggregate signature scheme," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 29, no. 3, pp. 303–310, 2017.
[54] H. Xiong *et al.*, "Heterogeneous signcryption with equality test for IIoT environment," *IEEE Internet Things J.*, early access, Jul. 13, 2020, doi: 10.1109/JIOT.2020.3008955.

Yingzhe Hou received the B.S. degree from Northeast Forestry University, Harbin, China, in 2018. She is currently pursuing the M.S. degree with the School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu, China.

Her research interests include equality test and signcryption.

Hu Xiong received the Ph.D. degree from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 2009.

He is currently a Full Professor with UESTC. His research interests include public-key cryptography and networks security.

Xin Huang received the B.S. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2019, where he is currently pursuing the M.S. degree with the School of Information and Software Engineering.

His research interests include equality test and identity-based public-key cryptography.

Saru Kumari received the Ph.D. degree in mathematics from Chaudhary Charan Singh University, Meerut, India, in 2012.

She is currently an Assistant Professor with the Department of Mathematics, Chaudhary Charan Singh University. She has authored or coauthored more than 190 research papers in reputed international journals and conferences, including 170 publications in SCI-indexed journals. Her current research interests include information security and applied cryptography.

Dr. Kumari was a Lead/Guest Editor of four Special Issues in SCI journals of Elsevier, Springer, and Wiley. She is on the editorial board of more than a dozen of international journals of high repute, including SCI-journals under Elsevier, Springer, and Wiley. She has been on the Advisory Committee and Technical Program Committee for many international conferences.

Authorized licensed use limited to: NANJING NORMAL UNIVERSITY. Downloaded on December 07,2022 at 10:48:34 UTC from IEEE Xplore. Restrictions apply.