Fully Secure Lightweight Certificateless Signature Scheme for IIoT

FATEMEH REZAEIBAGHA\textsuperscript{2}, YI MU\textsuperscript{1}, (SENIOR MEMBER, IEEE), XINYI HUANG\textsuperscript{1}, WENJIE YANG\textsuperscript{1}, KE HUANG\textsuperscript{3}

\textsuperscript{1}Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Mathematics and Informatics, Fujian Normal University, Fuzhou, Fujian, China (e-mail: ymu.ieee@gmail.com; xyhuang81@yahoo.com; njmuwyj@163.com)

\textsuperscript{2}School of Engineering and Information Technology, Murdoch University, Perth, Australia (e-mail: Fateem.Rezaeibagha@murdoch.edu.au)

\textsuperscript{3}School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China (e-mail: kh936@uowmail.edu.au)

Corresponding author: Yi Mu (e-mail: ymu.ieee@gmail.com)

This work was partly supported by the National Natural Science Foundation of China (61872087, 61822202, 61872089, 61822202).

\title{ABSTRACT} In recent years, Industrial Internet of Things (IIoT) has become increasingly important for applications in the industry. Inevitably, security for IIoT has become a priority in order to deploy secure applications. Amongst available cryptographic tools, certificateless signature schemes offer sound authentication solutions and avoid public-key certification from Trusted Third Parties (TTP). Certificateless signatures solve the key escrow problem against the dishonest Private Key Generator (PKG) and has considered to be a useful tool for IIoT applications. Recently, Karati et al. (IEEE Trans. Industrial Informatics, vol.14, no. 8, 2018) presented a lightweight certificateless signature scheme for IIoT Environments. This scheme was then broken by Zhang et al. (IEEE Access, vol. 8, 2018) by simply allowing to change the public key of the signer and using the homomorphic property of the original scheme. In this paper, we introduce a new attack to the scheme against the existential unforgeability, which is universal since we do not have to assume homomorphic property. We then introduce an entirely new lightweight certificateless signature scheme, which has been proven to be fully secure against all attacks found earlier. Our scheme is the first lightweight certificateless signature scheme with full security and is the most efficient in comparison with other existing schemes. It is desirable for IIoT applications. We also provide experimental results to justify our claims.

\title{INDEX TERMS} IIoT authentication, certificateless signature, provable security

\section{I. INTRODUCTION} The application of IoT across various industrial sectors is called Industrial Internet of Things (IIoT) which enables automation using cloud computing, such as manufacturing, transportation, energy management, smart healthcare, etc. [1]. Compared with the traditional industrial monitoring systems, IIoT increases flexibility and intelligent processes that lower down costs of development and maintenance. While emerging IIoT offers new opportunities, it poses new security threats related to unauthorised data access.

In IIoT, an enormous amount of data collected from different sources is needed to be processed and analysed. Cloud servers are used to ease data exchange and computation between sensors, smart devices and users due to the constrained feature of IIoT devices. For example, cloud-assisted IIoT is one of the applications for providing health monitoring [2]. However, cloud data access arises new challenges because data can be accessed, modified or deleted by unauthorised users. Hereby, many authentication protocols and digital signature schemes have been proposed in the literature to ensure data authenticity. In addition, energy consumption is an important criteria in designing security protocols for IIoT devices because such security implementations require expensive cryptographic operations. Therefore, it is important to design lightweight and secure cryptographic schemes to reduce energy consumption.

In traditional signature schemes, the binding between a user and his public key is provided by a certificate issued by a trusted third party (TTP) (or certification authority (CA)). To simplify the certificate management process, in 1984, Shamir [3] introduced the notion of identity-based cryptography (IBC) where the user’s public key is his identity (such as email address, name, etc.) as some unique information. The first practical identity-based encryption (IBE) was proposed by Boneh and Franklin in 2001 [4]. In their scheme, the TTP or key generation center (KGC) is completely trusted and
holds the user’s private key which can then generate the secret keys of all its users. It is also a concern that the private keys of users must be distributed over secure channels [5]. Then, the notion of certificateless public key cryptography (CL-PKC) was introduced by Al-Riyami and Paterson [6] in Asiacrypt 2003 to avoid the key-escrow of IBC.

In CL-PKC, certificates are no longer needed for ensuring the authenticity of public keys and therefore the key escrow problem in IBC is alleviated. However, similar to IBC, CL-PKC relies on the existence of a KGC who holds the "master-key". To elaborate, KGC does not hold private key of the users, yet it can produce partial private key $D_i$. The partial private key is constructed from the user’s identity $ID_i$, and then is transferred to the user $i$ via a secure channel. Consequently, the user $i$ computes his real private key by combining his partial private key $D_i$, some secret information, and the KGC’s public parameters to generate his private signing key and the corresponding public verification key.

There are two types of attacks against CL-PKC: Type-I - Key Replacement Attack [7] replaces the user’s public key with a chosen value, without having access to $D_i$. The second type of attack (Type-II - Malicious KGC) refers to a KGC who knows $D_i$ and impersonates the user, but it does not know the user’s secret key or cannot replace the public key [7], [8].

In a recent work, Karati et al. [9] introduced a certificateless signature (CLS) scheme for IIoT environments. Unfortunately, it has been broken by Zhang et al. [10] showing that it cannot resist both Type-I and Type-II attacks. Because of the homomorphic property of the CLS scheme of Karati et al., Zhang et al. also showed that it is not existentially unforgeable. In fact, with two valid signatures obtained from signature queries, one can obtain a new signature on an message $m = m_i m_j$, where $m_i$ and $m_j$ are associated with two different signatures obtained by the adversary but $m$ might not be meaningful to the verifier. They called it "known message attack". Zhang et al. also showed that the security proofs of the work proposed by Karati et al. are incorrect.

In this paper, we show that the known message attack can be easily repaired by hashing the message. However, we also found a universal attack against Karati et al.’s signature, even if the known message attack by Zhang et al. has been fixed. Our attack shows that any one who has a valid signature from signature queries can forge a new signature on an arbitrary message; therefore, we call it "universal attack". We present an entirely new lightweight CL signature scheme which is secure against all identified attacks and provide completely security proofs. We also carried out an experiment to simulate our scheme and other related schemes and show that our scheme is the most efficient.

II. RELATED WORK
Al-Riyami et al. [6] introduced the notion of certificateless signature (CLS) and encryption (CLE) in 2005. We refer them as certificateless cryptography (CL-PKC). Since then, several CLS schemes were proposed to improve the original scheme and security models. In [11], a generic construction of CLS was presented. Kang et al. [12] proposed a CLS scheme based on bilinear pairings and proved its security in the random oracle model assuming that the underlying group is Gap Diffie-Hellman group, to which the security proof is reduced. They borrowed the idea of ID-based signature and BLS signature and created a delegation-by-certificate proxy signature scheme which enables delegation of signing rights. These early works did not consider key replacement attacks.

To improve the CL-PKC scheme, Al-Riyami proposed a new CL-PKC scheme [13]. They provide a generic conversion to obtain a CBE scheme from any secure CL-PKC scheme which improves the original Gentry’s CBE scheme [14]. Based on this work, Huang et al. [7] showed that the proposed Al-Riyami’s CL-PKC is insecure in their security model. The reason is that, an attacker without possessing the master key can launch a public key replacement attack. In the same paper, Huang et al. [7] provided a new CLS scheme to repair the problem. Later, Au et al. [8] presented new CLS security models which assume that the KGC is passive and cannot actively replace the user public key or corrupt the user secret key. They therefore allow a malicious KGC with some conditions. Huang et al. [15] revisited security models of the CLS scheme and proposed new constructions in the random oracle model. The paper introduced three kinds of adversaries, namely Normal Adversary, Strong Adversary, and Super Adversary. However, the proposed scheme by Huang et al. is proved to be universally forgeable by Type-I adversary who can replace users’ public keys and generate new signatures under the replaced public keys [16]. To enhance the proposed CLS schemes in real applications (e.g. IIoT), several works proposed short CLS schemes (e.g., [17]–[19]). The idea of the short CLS schemes is to use signature schemes in low-bandwidth communication devices.

Recently, Karati et al. [9] proposed a lightweight certificateless signature scheme for IIoT environments. Unfortunately, it has been broken by Zhang et al. [10], who showed it is insecure against Type-I attacks and Type-II attacks. Zhang et al. [10] also showed that security proofs in Karati et al. [9] are incorrect. Based on these work we provide a comprehensive analysis and solution to the CLS scheme, which has a special significance to the research and application in IIoT.

III. SYSTEM MODEL
We consider a general IIoT scenario that consists of a key generation center (KGC), a cloud server which provides storage service for IIoT users, a data owner and a data user who is usually not a data owner, magenta and IIoT sensors. The system model of our improved IIoT CLS scheme consists of the following four entities:

- **Private Key Generator (KGC):** generates system public parameters and partial-private-keys for cloud server and users (data owner and data user).
- **Cloud Server:** responsible for processing data from data owners. It then communicates with users for data exchange and computation.
Properties:

- Data Owner: creates certificateless signature (CLS) scheme of his data with his private signing key which is generated with the partial private key received from KGC. The corresponding public key is sent to data user for signature verification. The CLS data is stored on the cloud server and therefore data user retrieves the verification.
- Data User: receives his partial-private-key from KGC and data owner’s public key to verify the CLS scheme. With the partial private key, he can generate his own private signing key and its corresponding public key. For authentication, he can also sign his communication flows for communication with the cloud.

As shown in Figure 1, in the proposed IIoT system, users are initially needed to be registered with KGC who is considered as a network administrator. The KGC generates and publishes public parameters and partial private keys. Users send their identities for registration to KGC and receive their partial private keys. Then, data owners and data users create their private signing keys and use them for generating CLS of IIoT data. The user’s keys are stored on their smart IIoT devices. The IIoT data from IIoT sensors are stored on the cloud server and then collected by the data owner, signed and stored as CLS data. Finally, the stored IIoT data on the cloud can be retrieved by data users and verified with the given data owner’s public key.

In the rest of presentation, we revisit and analyse the security of Karati et al.’s scheme and its security issues. Then, we propose an entirely new CLS which offers full security against all identified attacks and present a complete security proof using the standard security model (without random oracles). Because of its lightweight security and efficiency, it can be adopted in our proposed IIoT scenario.

IV. DEFINITIONS

A. BILINEAR PAIRING

Let $G$ and $G_T$ be cyclic group pairs of prime order $p$ and $e : G \times G \rightarrow G_T$ be a bilinear mapping with the following properties:

- Bilinearity: $e(g^a, g^b) = e(g, g)^{ab}$ for all $g \in G$ and $a, b \in \mathbb{Z}_p^*$.

- Non-degeneracy: $e(g, g) \neq 1_{G_T}$.

- Computability: The map $e$ can be computed efficiently.

B. $q$-BSDH ASSUMPTION

Let $G$ be an ordered cyclic group of prime order $p$ with generator $g$. The $q$-BSDH is stated as follows: Given as input a $(q + 1)$-tuple

$$\langle g, g^x, g^{x^2}, \cdots, g^{x^q} \rangle \in G^{q + 1},$$

output a pair

$$\langle c, e(g, g)^{c + x}\rangle \in \mathbb{Z}_p \times G \text{ for } c \in R \mathbb{Z}_p \setminus \{x\}.$$

$A$ solves the $q$-BSDH problem in $G$ with advantage $\epsilon$, if

$$\text{Adv}_{q}^\text{BSDH} = \Pr[ A(g, g^x, g^{x^2}, \cdots, g^{x^q}) = \langle c, e(g, g)^{c + x}\rangle ] \geq \epsilon$$

where the probability is over the random choice of generators $g \in R G$, the random choice of $x \in R \mathbb{Z}_p$, and the random bits consumed by $A$.

Definition 1 ($q$-BSDH assumption): The $(q, t, \epsilon)$-BSDH assumption holds in $G$, if no $t$-time algorithm has advantage at least $\epsilon$ in solving the $q$-BSDH problem in $G$.

C. FORMAL DEFINITION OF CLS

The formal structure of a CLS scheme consists of the following seven algorithms:

1) $(\text{msk}, \text{params}) \leftarrow \text{Setup}(\lambda)$. Given as input security parameter $\lambda$, the algorithm outputs the master key $\text{msk}$ and the system public parameters $\text{params}$.

2) $D_i \leftarrow \text{Set-Partial-Private-Key}(\text{msk}, I D_i)$. Given as input master key $\text{msk}$ and user’s identity $I D_i$, the algorithm outputs private key $D_i$.

3) $x_i \leftarrow \text{Set-Secret-Value}(\text{params}, I D_i)$. Given as input a parameter list $\text{params}$, partial private key $D_i$, the algorithm outputs secret value $x_i$.

4) $S K_i \leftarrow \text{Set-Secret-Key}(\text{params}, x_i, D_i)$. Given as input public parameters $\text{params}$, the user’s partial private key $D_i$ and the user’s secret value $x_i$, the algorithm outputs a private signing key $S K_i$.

5) $Y_i \leftarrow \text{Set-Public-Key}(x_i, I D_i)$. Given as input secret value $x_i$ and the user’s identity $I D_i$, the algorithm outputs the user’s public key $Y_i$.

6) $\sigma \leftarrow \text{CLS-Sign}(S K_i, m)$. Given as input signer’s secret key $S K_i$ and message $m \in \{0, 1\}^*$, the algorithm outputs CLS signature $\sigma$.

7) True/False $\leftarrow \text{CLS-Verify}(\text{params}, I D_i, Y_i, m, \sigma)$. Given as input the public parameters $\text{params}$, the signer’s identity $I D_i$, the signer’s public key $Y_i$, message $m$, and signature $\sigma$, the algorithm outputs True if the signature $\sigma$ is valid. Otherwise, it outputs False.

D. ADVERSARIAL MODEL

Type-I: This type of adversary is referred to an adversary $A_1$ who does not have access to the master-key, but it has the ability to replace the public key of any user with a value chosen at random.
Type-II: This type of adversary is referred to an adversary \( A_{\text{II}} \) who has access to the master-key, but it does not have the ability to perform public keys replacement.

**Definition 2 (Type-I Adversary):** A CLS scheme is (\( t, q_{SP}, q_{SS}, q_s, \epsilon \)) Type-I secure under the adaptive chosen message and ID attack, if \( A_1 \) gains a negligible advantage against \( q_{SP} \) number of partial private key queries, \( q_{SS} \) number of set secret value queries, and \( q_s \) number of signature queries in polynomial time \( t \) in the below game between an adversary \( A_{\text{II}} \) and a challenger \( C \). \( \epsilon \) is the advantage of \( A_{\text{II}} \) to forge a signature.

- **Setup:** The challenger \( C \) runs the Setup algorithm to obtain system public parameters params and master key msk. Then, \( C \) sends params to the adversary \( A_{\text{II}} \).
- **Partial-Private-Key-Extract:** \( A_{\text{II}} \) requests the partial private key of the user with identity ID\( _1 \). Then, \( C \) outputs \( D_1 \) as the user’s partial private key.
- **Set-Secret-Value:** \( A_{\text{II}} \) receives the secret value \( x_i \) of identity ID\( _1 \) from \( C \).
- **Set-Secret-Key:** \( A_{\text{II}} \) receives the private key SK\( _i \) of ID\( _1 \) from \( C \).
- **Public-Key-Replace:** \( A_{\text{II}} \) set \( Y'_i \) as the new public key of the user and submit \((x_i, Y'_i, ID_1)\) to \( C \).
- **SignQueries:** \( A_{\text{II}} \) makes signature queries on message \( m \in M \) wrt any ID, where \( m \neq m^* \) for ID\( ^* \). In response, \( C \) outputs a valid signature \( \sigma(m) \) under the public key of \( A_{\text{II}} \).
- **Output:** Finally, \( A_{\text{II}} \) outputs a message and signature pair \( (m^*, \sigma^*) \) of the user with identity ID\( ^* \). The message and signature pair \( (\sigma^*, m^*) \) must satisfy the following conditions:
  - \( A_{\text{II}} \) has not queried Set-Partial-Private-Key and Sign of the tuple \((ID^*, Y^*, m^*)\).
  - The forged signature \( \sigma^* \) is valid under the public key \( Y^* \) chosen by \( A_{\text{II}} \), where \( A_{\text{II}} \) may change the public key.

**Definition 3 (Type-II Adversary):** A CLS scheme is (\( t, q_{SP}, q_{SS}, q_s, \epsilon \)) Type-II secure under the adaptive chosen message and ID attack, if \( A_{\text{II}} \) gains a negligible advantage against \( q_{SS} \) number of set secret value queries, and \( q_s \) number of signature queries in polynomial time \( t \) in the following game between an adversary \( A_{\text{II}} \) and a challenger \( C \). \( \epsilon \) is the advantage of \( A_{\text{II}} \) to forge a signature.

- **Setup:** The challenger \( C \) runs the Setup algorithm to obtain public parameters params and the master key msk. Then, \( C \) sends params and msk to the adversary \( A_{\text{II}} \).
- **As in the definition of Type-I adversary:** \( A_{\text{II}} \) makes Partial-Private-Key-Extract queries, Set-Secret-Value queries, Set-Secret-Key queries, Set-Public-Key queries, and SignQueries.
- **Output:** Finally, \( A_{\text{II}} \) outputs a signature \( \sigma^* \) on message \( m^* \) of the user with identity ID\( ^* \). The message and signature pair \((\sigma^*, m^*)\) must satisfy the following conditions:
  - \( A_{\text{II}} \) has not queried Set-Secret-Value and Sign of the tuple \((ID^*, Y^*, m^*)\).
  - The forged signature \( \sigma^* \) is valid under the public key \( Y^* \) chosen by \( A_{\text{II}} \).

**V. REVIEW OF KARATI ET AL.’S SCHEME**

The proposed CLS scheme in [9] consists of the following six algorithms:

- **params \( \leftarrow \) Setup(\( \lambda \)).** Taking as input security parameter \( \lambda \), KGC outputs system parameters params = \( \langle G, G_T, g_1, g_2, e, p, \text{YKOC}, H \rangle \), where \( G = \langle g_1 \rangle \) is a group of prime order \( p \) generated by \( g_1 \), and a bilinear pairing \( e : G \times G \rightarrow G_T \). KGC selects a hash function \( H : \{0, 1\}^* \rightarrow G^* \) and master secret key \( \text{msk} = y \in R Z_p^* \). KGC sets \( g_2 = e(g_1, g_1)^y \) and public key \( \text{YKOC} = g_1^y \). KGC keeps \( \text{msk} = y \) as private key and outputs public parameters params.
- **(\( D_1, h_i, y_i \) \( \leftarrow \) Set-Partial-Private-Key(params, msk, ID\( _1 \)).** Taking as input public parameters params, master private key \( y \) of KGC, and user’s identity ID\( _1 \), it computes \( h_i = H(ID_1) \) and \( y_i = g_1^{k_iy + r_i} \) where \( r_i \in R Z_p^* \). Then, KGC computes the partial private key \( D_i = (y_i, R_i) \) where \( R_i = g_1^{y_i} \). Accept the partial private key \( D_i \) if the following equation holds
  \[
eq e(g_1, \text{YKOC})^{h_i} = e(y_i, (g^{h_i} \cdot R_i \cdot \text{YKOC})^y) \]
- **SK\( _i \) \( \leftarrow \) Set-Secret-Value(ID\( _1 \)).** Given public parameters params, user’s identity ID\( _1 \), the algorithm computes the secret key \( \text{SK}_i = (c_i, x_i, R_i) \) where \( (c_i, x_i) \) are chosen at random.
- **Y\( _i \) \( \leftarrow \) Set-Public-Key(x\( _i \), ID\( _i \)).** Given public parameters params and secret value \( x_i \), the algorithm returns the user public key \( Y_i \) as follows:
  \[
Y_i = (Y_{i1}, Y_{i2}) = \left( \frac{1}{y_i}, c_i \right) \]
- **\( \sigma \) \( \leftarrow \) CLS-Sign(params, SK\( _i \), m).** Given public parameters params, secret key \( \text{SK}_i \), and message \( m \in Z_p^* \), it selects \( t \in R Z_p^* \) and computes \( h_i, \sigma_1, \) and \( \sigma_2 \) as follows
  \[
h_i = H(ID_1),
\]
  \[
\sigma_1 = g_2^{t},
\]
  \[
\sigma_2 = \left( g_1^{h_i} \cdot R_i \cdot \text{YKOC} \right)^{\left( \frac{\sigma_i}{\sigma_1} - t \right) x_i}.
\]
It outputs signature of message \( m \) as \( \sigma = (\sigma_1, \sigma_2) \).
- **Valid/Invalid \( \leftarrow \) CLS-Verify(params, ID\( _i \), Y\( _i \), m, \sigma).** Given public parameters params, signature \( \sigma \) of message \( m \), the user’s identity ID\( _1 \), and public key \( Y_i \). Outputs Valid if the following equation holds
  \[
eq e(Y_{i1}, \sigma_2) \]
Otherwise, outputs Invalid.
VI. OUR ATTACK ON KARATI’S CLS SCHEME

The security model in the work due to Karati et al. [6] captures the existentially unforgeability against Type-I and Type-II adversaries. Zhang et al. [10] showed that the scheme of Karati et al. is actually forgeable under the attacks of Type-I or Type-II adversaries. Their Type-II attack can be fixed by changing the message $m$ to a hash message. In this section, we show that the scheme proposed by Karati’s et al. is insecure even if the attacks in Zhang et al.’s scheme had been fixed. In our attack, the adversary can always forge a user’s signature on a message of its choice; therefore it is universal.

Here, we show only CLS-Sign algorithm. The rest of the algorithms are same as described in the original scheme in [9].

**CLS-Sign:** The adversary received a valid signature $\sigma(m)$ of ID$_i$ in the signature query list from the challenger $C$ where $m \in \mathbb{Z}_p^*$. Ignoring the signature challenge from $C$, to forge the signature, the adversary $A_i$ selects a message $m’ \in \mathbb{Z}_p^*$ where $m’ \neq m$ and $m’$ could be the challenge message $m^*$, and then performs the following to compute the forged signature of ID$_i$:

- Compute $\sigma_1$ as
  $$\sigma_1’ = (g_i^{r_1})^{m’} = g_i^{t’},$$
  where $t’ = t \cdot \frac{m}{m’}$.
- Compute $\sigma_2$ as
  $$\sigma_2’ = \left( g_i^{r_1} \cdot R_i \cdot Y_{KGC} \right) \left( \frac{m’}{m} \right)_{x_i},$$
  $$\sigma_2’ = \left( g_i^{r_1} \cdot R_i \cdot Y_{KGC} \right) \left( \frac{m’}{m} \right)_{x_i}.$$
- Publish $\sigma’ = (\sigma_1’, \sigma_2’)$ as the forged signature of message $m’$.

The attack is successful as it is easy to find that the verification on $(\sigma’, Y_i)$ returns Valid.

Compared with the known message attack, our attack gives a valid signature on any meaningful message, even if the message is hashed before signed. Therefore, the fix to the known message attack cannot be applied to our attack. Actually, our attack can be easily repaired by simply changing $m$ to $H(\sigma_1, m)$. However, there are still remaining problems: (1) the key replacement attacks still stand; (2) the scheme cannot be proved due to the challenger cannot answer the signature queries when ID = ID’ and $m \neq m’$; (3) the reduction in the proofs is based on the public key, which is entirely wrong. Therefore, their scheme is beyond repairable.

VII. OUR FULLY SECURE LIGHTWEIGHT CLS SCHEME

A. THE SCHEME

params ← Setup($\lambda$). Taking as input security parameter $\lambda$, KGC outputs system parameters params = $\langle G, G_T, g, g_1, e, p, Y_{KGC}, H \rangle$, where $G$ and $G_T$ are multiplicative groups of prime order $p$, and a bilinear pairing $e : G \times G \rightarrow G_T$. KGC selects a hash functions $H : \mathbb{Z}_p \rightarrow \mathbb{Z}_p$ and $H_1 : \{0, 1\}^* \rightarrow G$ and the master secret key $msk = s \in \mathbb{Z}_p$. KGC’s public key is $Y_{KGC} = g^s$. KGC outputs public parameters params.

$$d_i \leftarrow \text{Set-Partial-Private-Key}(\text{params, } msk, \text{ID}_i).$$
Taking as input public parameters params, master private key $s$ of KGC, and user’s identity $\text{ID}_i$. It computes $d_i = g^s$, and $Y_{i1} = g_1$, where $g_1 = H_1(\text{ID}_i) \in \mathbb{Z}_G$ and is unique for user $\text{ID}_i$. $d_i$ is the private key for user $\text{ID}_i$. Accept, if $e(d_i, g) = e(g_1, Y_{KGC})$.

$$SK \leftarrow \text{Set-Secret-Value}(\text{ID}_i).$$
Given public parameters params, user’s identity $\text{ID}_i$, it outputs the secret key $SK_i = x_i$ where $x_i$ is chosen at random.

$$Y_i \leftarrow \text{Set-Public-Key}(x_i, \text{ID}_i).$$
Given public parameters params, secret value $x_i$ and identity $\text{ID}_i$, it outputs the user public key $Y_i$ as $Y_{i1} = g_1, Y_{i2} = g^{x_i}$. Check if the equality holds: $e(d_i, g) = e(g_1, Y_{KGC})$.

$$\sigma \leftarrow \text{CLS-Sign}(\text{params, } SK_i, d_i, m).$$
Given public parameters params, secret key $SK_i$, $d_i$, and message $M \in \mathbb{Z}_p$, it selects $R \in \mathbb{Z}_p$ and computes

$$S = d_i^{r_1 H(r_1, m’; x_i)} = g_1^{r_1 H(r_1, m’; x_i)}.$$ 

It outputs the signature of message $M$ as $\sigma = (r, S)$.

Valid/Invalid ← CLS-Verify(\text{params, } \text{ID}_i, Y_i, m, \sigma).$$
Given public parameters params, signature $\sigma$ of message $M$, the user’s identity $\text{ID}_i$ and public key, check

$$e(Y_{i1}, Y_{GKC}) = e(S, (g^{r \cdot (Y_{i2})^H(r, Y_{i2}, m)})).$$

The correctness is obvious:

$$e(Y_{i1}, Y_{GKC}) = e(S, (g^{r \cdot (Y_{i2})^H(r, Y_{i2}, m)})),$$
$$= e\left( g_i^{r H(r, Y_{i2}, m; x_i)} \cdot g^{r \cdot H(r, Y_{i2}, m; x_i)} \right),$$
$$= e\left( g_i^{r H(r, Y_{i2}, m; x_i)} \cdot g^{r H(r, Y_{i2}, m; x_i)} \right),$$
$$= e(g_1, g),$$
$$= e(H_1(\text{ID}_i), g^s),$$
$$= e(Y_{i1}, Y_{GKC}).$$

B. SECURITY PROOF

**Theorem 1:** Suppose the $(q, t’, \epsilon)$-BSDH assumption holds in $G$. If there exists a PPT algorithm $A_t$ that can break $(t, t_{pp}, q_s, q_s, \epsilon)$-secure against existential forgery including key replacement with advantage $Adv_{A_t}^{q_{BSDH}}$, then there exists a PPT algorithm $C$ that can solve the $q$-BSDH problem with advantage $Adv_{C}^{q_{BSDH}} \geq Adv_{A_t}^{q_{BSDH}}(\lambda) \geq \left(1 - \frac{q_{pp} + q_s + q_s}{p}\right) \cdot \frac{1}{p} \epsilon$. where $T$ is the maximum time for an exponentiation in $G, G_T$, and $\mathbb{Z}_p$.

**Proof.** For simplicity of presentation, we omit the hash function $H()$. Suppose that there is an adversary $A_t$ that can
break Type-I security of our proposed CLS scheme. Given the $q$-BSDH problem instance $T = (G, g, g^x, g^{x^2}, \cdots, g^{x^{q-1}})$, the challenger $C$ uses $A_i$'s queries and responses to compute the solution $(c, e(g, g)^{tx})$ of the $q$-BSDH instance for some known $c \in_R Z_p^*$, $c \in Z_p \backslash \{x\}$ and unknown $x \in R Z_p^*$, where $G$ is a multiplicative group of order $p$. We assume $ID^*, m^*$ cannot be queried together by $A_i$.

Consider $A_i = g^{x_i^*}$ where $\forall i \in [1, q]$. The algorithm $C$ proceeds the following game with the adversary $A_i$. Assume $ID^*, m^*$ cannot be queried together by $A_i$.

Setup : Challenger $C$ creates a list $L = (ID, Y_1, Y_2, x, d)$ that is empty in the beginning. $C$ selects private key $s \in R Z_p^*$ and computes public key $Y_{KGC} = g^s$. $C$ sends public parameters $(G, GT, e, g, p, Y_{KGC}, H, H_1)$ to $A_i$.

Create-User $ID_i : C$ sets $h_i = H(ID_i)$ and creates user ID_i and add $(ID_i, *, *, *, *)$ to the list $L$, which initially is empty.

Partial-Private-Key-Extract : $A_i$ sends $qs_p$ partial private key queries on $ID_i$ to $C$, where $ID_i \neq ID^*$. If $ID_i$ is not in the list, $C$ calls Create-User for new ID_i. Otherwise, $C$ returns $d_i$ if it is in the list $L$. If $d_i$ is not in the list $L$, $C$ computes $d_i = g_{s_i}$, where $Y_{1i} = g_{s_i} \in G$ is set for $ID_i$ only. $C$ adds $(ID_i, Y_{1i}, *, *, d_i)$ to the list $L$. If $ID_i = ID^*, C$ aborts.

Set-Secret-Value : $A_i$ sends $qs_s$ queries to $C$. For $ID_i \neq ID^*$ if $ID_i$ and $x_i$ are in the list $L$, $C$ returns $x_i$. If $x_i = *$, $C$ selects $x_i \in R Z_p^*$ as new secret value for $ID_i$ and adds $(ID_i, Y_{1i}, *, x_i, d_i)$ to the list $L$. If $ID_i$ is not in the list, it calls Create-User to create the new user ID_i and adds $x_i$ to the list $L$. If $ID_i = ID^*, C$ is unable to provide $x_i$ and aborts.

Set-Public-Key : $A$ sends $qs_p$ queries on a selected ID_i to $C$. For $ID_i \neq ID^*$ if $ID_i$ is in $L$, $C$ returns $Y_i = (Y_{1i}, Y_{2i})$, where $Y_{1i} = g_{s_i}, Y_{2i} = g^{x_i}$, and update $L = (ID_i, Y_{1i}, Y_{2i}, x_i, d_i)$. Otherwise, if $ID_i = ID^*, C$ returns: $Y_{1i} = g_{s_i}, Y_{2i} = g^s$, where $g^s$ is given in the $q$-BSDH instance, and updates the list $L = (ID_i, g_{s_i}, g^s, *, d_i)$.

Sign-Queries : Adversary $A_i$ makes $qs$ signature queries for $(ID_i, m_j)$ to $C$. If $ID_i \neq ID^*$ and $x_i$ is in the list $L$, $C$ selects $r_j \in R Z_p^*$ and computes a signature easily as follows,

$$\sigma = \frac{1}{S_{ij}} = g_{t_j^*}^{x_i^*},$$

where $t_j = H(r_j, Y_{2i}, m_j)$. $C$ outputs the signature $\sigma = (r_j, S_{ij})$ and sends it to $A_i$.

If $ID_i = ID^*$ and $m_j \neq m^*$, where $x_i = *$, $C$ performs the following:

Let $f$ be the univariate polynomial defined as

$$f(X) = \prod_{i=1}^{q} (X + r_i).$$

Expand $f$ and write $f(X) = \sum_{i=0}^{q} \alpha_i X^i$, where $\alpha_0, \cdots, \alpha_q \in Z_p$ are the coefficients of the polynomial $f$. $C$ picks a random $\theta \in Z_p$ and computes $g_s = \prod_{i=0}^{q} A_i^\theta$. Let $f_j$ be the polynomial

$$f_j(X) = f(X)/(X + r_j).$$

We expand $f_j$ and rewrite $f_j(X) = \sum_{i=0}^{q} \alpha_i X^i$. $C$ then computes

$$S_{ij} = \prod_{i=0}^{q-1} A_i^{\theta_s} + g^\theta_s f_j(t_j, x) = g^{\gamma_{s_j} x^t_j},$$

where $C$ has rewritten $x$ as $t_j \cdot x$. Returns $\sigma = (r_j, S_{ij})$ as a valid signature on $m_j$.

We can easily find that $\sigma$ is a valid signature on message $m_j$ under the public key $Y_{s_1} = g_s, Y_{s_2} = g^x$. $A_i$ can continue signature queries before returning the forged signature.

Public-Key-Replacement : $A_i$ replaces the public key $Y_{s_1} = g_s, Y_{s_2} = g^x$ with $Y'_{s_1} = (Y'_{s_1}, Y'_{s_2})$, and forwards to $C$.

Forgery : Finally, $A_i$ outputs a forged signature $\sigma = (r^*, S^*)$ for a message $m^* \notin \{m_1, \cdots, m_q\}$ where $ID^* \neq ID_i$ with public key $Y_{s_1}'$ or $Y_{s_2}'$. If the $\sigma$ can be verified with $X_{ID^*}'$, $A_i$ wins the game and $C$ is unable to solve the $q$-BSDH problem. However, it is negligible in our game, as a PPT $A_i$ is not able to construct a forged signature for any replaced public key $Y_{ID^*}'$. The successful forgery should have

$$S^* = g_{s^*}^{\gamma_2 X^t_2}.$$ The forgery is successful if $CLS$-Verify (params, $m^*, \sigma^*, ID^*, Y_{ID^*}')$ outputs Valid, otherwise $C$ aborts.

The correctness:

$$e(Y_{s_1}, Y_{KGC}) = e(S^*, (g^{x^*}, (Y_{2i}^t)^*)).$$

$$= e(g^{x^* x^*}, (g^{x^*} \cdot g^x))$$

$$= e(g^{x^* x^*}, g^{x^*+t^* x})$$

$$= e(g^{x^*}, g)$$

$$= e(H_1(ID^*), g^s)$$

$$= e(Y_{s_1}, Y_{KGC}).$$

Then it rewrites

$$S^* = g^{\theta_s f_j(x^t)} = g^{\theta_s h(x^t)}.$$ (1)

where $x^t = t \cdot x$.

$C$ uses the long division to compute $f(x)/(x + r^*)$ and computes polynomial $f(X)$ as $f(X) = (X + r^*) \gamma(X) + \gamma^*$, where $\gamma(X)$ is a computable polynomial $\gamma(X) = \sum_{i=0}^{q-1} \gamma_i X^i$ and $\gamma^* \in Z_p$ is a constant. Then, it computes $f(X)$ as follows:

$$f(X)/(X + r^*) = \frac{\gamma^*}{X + r^*} + \sum_{i=0}^{q-1} \gamma_i X^i,$$

and can deduce $S^*$ from equation (1) as

$$S^* = g^{\theta_s ((\gamma_{s_j}^2 x^t_2 + \sum_{i=0}^{q-1} \gamma_i x^t_2))}.$$
where $\gamma^* \neq 0$. Notice that since $f(X) = \prod_{i=1}^{q} (X + r_i)$ and $m^* \notin \{m_1, \ldots, m_q\}$, $f(X)$ cannot be divided by $(X + r^*)$ and therefore $C$ outputs the solution $e(g_1, g_1)_{r^*/r}$ with the forged $S^*$ as follows:

$$W = \left( e(g, (S^*)^{1/s}, \prod_{i=0}^{q-1} g^{-\gamma_i^{*r^*}} \right)^{1/\gamma^*}$$

$$= \left( e(g, g^{\frac{r^*}{r+\gamma^*}} \cdot \prod_{i=0}^{q-1} g^{-\gamma_i^{*r^*}} \right)^{1/\gamma^*}$$

$$= e(g, g)^{1/(\gamma^*+r^*)}$$

$C$ outputs the solution $(c, e(g, g)^{\gamma^*/r})$ where $c = m^*/r^*$, and it then breaks the $q$-BSDH assumption.

In the following, we perform a probability analysis. For a successful forgery, we consider three events:

- $\Gamma_1$: $C$ does not abort during the simulation.
- $\Gamma_2$: $S^*$ is a valid forged signature on $m^*$ for $ID^*$. 
- $\Gamma_3$: The advantage of $A_H$ to forge a signature.

The total probability for $\Gamma_1$ is

$$\Gamma_1 = \left( 1 - \frac{q_{pp}}{p} \right) \left( 1 - \frac{q_{ss} + q_s}{p} \right) \geq \left( 1 - \frac{q_{pp} + q_{ss} + q_s}{p} \right).$$

For Forgery, $A_H$ outputs a forged signature. If the verification outputs Valid, then the game ends; otherwise $C$ aborts. The probability $C$ does not abort is $\Pr[\Gamma_2] = 1/p$.

Assume the advantage that $A_H$ forges a signature successfully is $\Pr[\Gamma_3] \geq \epsilon$. The overall success probability of breaking $q$-BSDH assumption is:

$$\Pr[\Gamma_1 \land \Gamma_2 \land \Gamma_3] = \Pr[\Gamma_1] \cdot \Pr[\Gamma_2] \cdot \Pr[\Gamma_3].$$

The advantage of $A_H$ breaking the $q$-BSDH assumption is:

$$\text{Adv}_{A_H}^{\text{BSDH}}(\lambda) \geq \left( 1 - \frac{q_{pp} + q_{ss} + q_s}{p} \right) \cdot \frac{1}{p} \cdot \epsilon.$$

We now compute the time complexity of the game. Let $T$ be the time cost for computing an exponentiation.

For Partial-Private-Key-Extract, $C$ requires $q_{pp} T$.

For Set-Public-Key, $C$ requires $q_{ss} T$.

For SignQueries, $C$ requires $(q_s + q_1) T$.

The total time cost is the sum of these time costs:

$$q_{pp} T + q_{ss} T + (q_s + q_1) T \approx \Theta(q_s + q_1) T.$$ 

The overall time for $A_H$ to break the $q$-BSDH assumption is considered as $t' \geq t + \Theta(q_s + q_1) T$.

**Theorem 2:** Suppose the $(q, t', \epsilon)$-BSDH assumption holds in $\mathbb{G}$. If there exists a PPT algorithm $A_H$ that can break $(t, q_{pp}, q_{ss}, q_s, \epsilon)$-secure against existential forgery with advantage $\text{Adv}_{A_H}^{\text{BSDH}}$, then there exists a PPT algorithm $C$ that can solve the $q$-BSDH problem with advantage

$$\text{Adv}_{C}^{\text{BSDH}} \geq \text{Adv}_{A_H}^{\text{BSDH}}(\lambda) \geq \left( 1 - \frac{q_{ss} + q_s}{p} \right) \cdot \frac{1}{p} \cdot \epsilon.$$

$t \leq t' - \Theta(q_s + q) T$,

where $T$ is the maximum time for an exponentiation in $\mathbb{G}, \mathbb{G}_T$, and $\mathbb{Z}_p$.

**Proof.** Suppose that there is an adversary $A_H$ that can break Type-II security of our proposed CLS scheme. Given the $q$-BSDH problem instance $T = (\mathbb{G}, g, g^a, \ldots, g^{a^t})$, the challenger $C$ uses $A_H$ queries and responses to compute the solution $(c, e(g, g)^{\gamma^*/r})$ of the $q$-BSDH instance for some known $c \in_R \mathbb{Z}_p^*$, $c \in \mathbb{Z}_p \setminus \{-x\}$ and unknown $x \in_R \mathbb{Z}_p^*$ where $\mathbb{G}$ is a multiplicative group of order $p$. We assume $ID^*$ and $m^*$ cannot be queried together by $A_H$.

Consider $A_i = g^{x^2}$ where $\forall i \in [1, q]$. The algorithm $C$ proceeds the following game with the adversary $A_H$. Assume $ID^*$ and $m^*$ cannot be queried together by $A_H$.

**Setup:** The challenger $C$ creates a list

$$L = (ID, Y_1, Y_2, x, d, h)$$

that is empty in the beginning. $C$ selects private key $s \in_R \mathbb{Z}_p^*$ and computes public key $Y_{soc} = g^s$.

$C$ sends public parameters

$$\text{params} = (\mathbb{G}, \mathbb{G}_T, c, g, p, Y_{soc}, H)$$

to $A_H$.

Create-User $ID_1 : C$ sets $h_i = H(ID_i)$ and creates user $ID_i$ and add $(ID_i, \star, \star, \star, \star, h_i)$ to the list $L$, which initially is empty.

**Partial-Private-Key-Extract :** $A_{II}$ sends $q_{pp}$ partial private key queries on $ID_i$ to $C$, where $ID_i \neq ID^*$ or $ID_i = ID^*$. If $ID_i$ is not in the list, $C$ calls Create-User for new $ID_i$. $C$ computes $d_i = g_i^s$, where $Y_1 = g_i \in \mathbb{G}$ is set for $ID_i$ only, and adds $(ID_i, Y_1, \star, \star, d_i, h_i)$ to the list $L$. Otherwise, $C$ returns $d_i$ if it is in the list $L$. If $ID_i = ID^*$, $C$ aborts.

**Set-Private-Value :** $A_{II}$ sends $q_{ss}$ queries to $C$. For $ID_i \neq ID^*$, $C$ returns $Y_s = (Y_1, Y_2)$ if $ID_i$ is in $L$, where $Y_1 = g_i$, $Y_2 = g^s$, and update $L = (ID_i, Y_1, Y_2, x_i, d_i, h_i)$. Otherwise, if $ID_i = ID^*$, $C$ returns:

$$Y_1 = g_s, \quad Y_2 = g^s,$$

where $g^s$ is given in the $q$-BSDH instance, and updates the list $L = (ID_i, Y_1, g^s, \star, \star, d_i, h_i)$.

**SignQueries :** Adversary $A_H$ makes $q_s$ signature queries for $(ID_i, m_j)$ to $C$. If $ID_i \neq ID^*$ and $x_i$ is in the list $L$, $C$ selects $r_j \in_R \mathbb{Z}_p^*$ and computes a signature easily as follows:

$$S_{ij} = d_i^{r_j \gamma_{ij}^*/r} = g_i^{r_j^{\gamma_{ij}^*/r}}.$$
where \( t_j = H(r_j, Y_{i2}, m_j) \). \( C \) outputs the signature \( \sigma = (r_j, S_{ij}) \) and sends it to \( A_\Pi \).

If ID\(_j\) = ID* and \( m_j \neq m^* \), where \( x_i = \ast \), \( C \) performs the following:

Let \( f \) be the univariate polynomial defined as

\[
f(X) = \prod_{i=1}^{q} (X + r_i).
\]

Expand \( f \) and write

\[
f(X) = \sum_{i=0}^{q} \alpha_i X^i,
\]

where \( \alpha_0, \cdots, \alpha_q \in \mathbb{Z}_p \) are the coefficients of the polynomial \( f \). \( C \) picks a random \( \theta \in \mathbb{Z}_p \) and computes \( g_* = \prod_{i=0}^{q} \alpha_i \cdot \theta \).

Let \( f_j(X) \) be the polynomial

\[
f_j(X) = f(X)/(X + r_j) = \prod_{i=1, i \neq j}^{q} (X + r_i).
\]

We expand \( f_j \) and rewrite \( f(X) = \sum_{i=0}^{q} \alpha_i X^i \). \( C \) then computes

\[
S_{ij} = \prod_{i=0}^{q-1} \alpha_i^{\ast \cdot s} = g^{\ast \cdot s} f_j(t \cdot x) = g^{x \cdot \text{rem}(t \cdot x)},
\]

where \( C \) has rewritten \( x \) as \( t \cdot x \). Returns \( \sigma = (r, S_{ij}) \) as a valid signature on \( m_j \).

We can easily find that \( \sigma \) is a valid signature on message \( m_j \) under the public key \( Y_{i1} = g_* \cdot Y_{i2} = g^\ast \). \( A_\Pi \) can continue signature queries before returning the forged signature.

For forgery: Finally, \( A_\Pi \) outputs a forged signature \( \sigma^* = (r^*, S^*) \) for a message \( m^* \notin \{m_1, \cdots, m_q\} \) where \( \text{ID}^* \neq \text{ID}_i \) with public key \( Y_{\text{ID}^*} \). The successful forgery should have

\[
S^* = (d_\ast)^{x \cdot \text{rem}(t^* \cdot x)},
\]

where \( d_\ast \) is the partial private key for \( \text{ID}^* \) and is held by \( A_\Pi \). The forgery is successful if CLS-Verify \( \text{(params, m*, \sigma*, ID*, Y_{ID*})} \) outputs Valid; otherwise \( C \) aborts.

The correctness:

\[
e(Y_{i1}, Y_{\text{ID}^*}) = e(S^*, (g^\ast \cdot Y_{i2})^{t^*})
\]

\[
= e(g_*^{x \cdot \text{rem}(t^* \cdot x)}, (g^\ast \cdot g^{t^* \cdot x}))
\]

\[
= e(g_*^{\ast \cdot \text{rem}(t^* \cdot x)}, g^{\ast \cdot t^* \cdot x})
\]

\[
= e(g^s, g)
\]

\[
= e(g, g^s)
\]

\[
= e(Y_{i1}, Y_{\text{ID}^*}).
\]

where \( h^* = H(\text{ID}^*) \). Then it rewrites

\[
S^* = g^{\frac{\delta(f(s \cdot x)) \cdot s}{x \cdot \text{rem}(t^* \cdot x)}} = g^{\frac{\delta(f(x \cdot s))}{x \cdot \text{rem}(t^* \cdot x)}}.
\]

where \( x' = t^* \cdot x \).

\( C \) uses the long division to compute \( f(x)/(x + r^*) \) and computes polynomial \( f(X) \) as

\[
f(X) = (X + r^*) \gamma(X) + \gamma^*
\]

where \( \gamma(X) \) is a computable polynomial \( \gamma(X) = \sum_{i=0}^{q} \gamma_i X^i \) and \( \gamma^* \in \mathbb{Z}_p \) is a constant. Then, it computes \( f(X) \) as follows:

\[
f(X)/(X + r^*) = \frac{\gamma^*}{X + r^*} + \sum_{i=0}^{q-1} \gamma_i X^i,
\]

and can deduce \( S^* \) from equation (2) as

\[
S^* = g^{\ast \cdot \text{rem}(x \cdot r^*) \cdot \sum_{i=0}^{q-1} \gamma_i x^i}.
\]

where \( \gamma^* \neq 0 \). Notice that since \( f(X) = \prod_{i=1}^{q} (X + r_i) \) and \( m^* \notin \{m_1, \cdots, m_q\} \), \( f(X) \) cannot be divided by \( (X + r^*) \) and therefore \( C \) outputs the solution \( e(g, g)^{\frac{1}{\text{rem}(x^r \cdot r^*)}} \) with the forged \( S^* \) as follows:

\[
W = e(g, g)^{\frac{1}{\text{rem}(x^r \cdot r^*)}} = e(g, g)^{\frac{1}{\text{rem}(x^r \cdot r^*)}}
\]

\[
= e(g, g)^{\frac{1}{x^r}}.
\]

\( C_\Pi \) rewrites

\[
W^{t^*} = e(g, g)^{t^* / (x^r \cdot r^*)} = e(g, g)^{\frac{1}{x^r}} = e(g, g)^{\frac{1}{x^r}}.
\]

\( C \) outputs the solution \((c, e(g, g)^{\frac{1}{x^r}}) \) where \( c = r^*/t^* \), and it then breaks the \( q \)-BSDH assumption.

In the following, we perform a probability analysis. For a successful forgery, we consider three events:

\( \Gamma_1: \) \( C \) does not abort during the simulation.

\( \Gamma_2: \) \( S^* \) is a valid forged signature on \( m^* \) for \( \text{ID}^* \).

\( \Gamma_3: \) The advantage of \( A_\Pi \) to forge a signature.

For Set-Secret-Value, the probability \( C \) does not abort is

\[
(1 - \frac{1}{p})^{q_\ast} \geq (1 - \frac{q_\ast}{p}).
\]

where \( 1/p \) is the probability \( A_\Pi \) asked the secret value \( x \) for user \( \text{ID}^* \).

For SignQueries, we have

\[
(1 - \frac{1}{p})^{q \ast} \geq (1 - \frac{q_\ast}{p}).
\]

The total probability for \( \Gamma_1 \) is

\[
\Gamma_1 = (1 - \frac{q_\ast}{p})(1 - \frac{q_\ast}{p}) \geq (1 - \frac{q_\ast + q_\ast}{p}).
\]

For Forgery, \( A_\Pi \) outputs a forged signature. If the verification outputs Valid, then the game ends; otherwise \( C \) aborts.

The probability \( C \) does not abort is \( \Pr[\Gamma_2] = 1/p \).
Assume the advantage that $\mathcal{A}_H$ forges a signature successfully is $\Pr[\mathcal{G}_3] \geq \epsilon$. The overall success probability of breaking $q$-BSDH assumption is:

$$\Pr[\mathcal{G}_1 \wedge \mathcal{G}_2 \wedge \mathcal{G}_3] = \Pr[\mathcal{G}_1] \cdot \Pr[\mathcal{G}_2] \cdot \Pr[\mathcal{G}_3].$$

The advantage of $\mathcal{A}_H$ breaking the $q$-BSDH assumption is:

$$\text{Adv}_{\mathcal{A}_H}^{\text{BSDH}}(\lambda) \geq \left(1 - \frac{q_{ss} + q_s}{p}\right) \cdot \frac{1}{p} \cdot \epsilon.$$

We now compute the time complexity of the game. Let $T$ be the time cost for computing an exponentiation.

For Set-Public-Key, $C$ requires $q_{sp}T$.

For SignQueries, $C$ requires $(q_s + q)T$.

The total time cost is the sum of these time costs:

$$q_{sp}T + (q_s + q)T \approx \Theta(q_s + q)T.$$

The overall time for $\mathcal{A}_H$ to break the $q$-BSDH assumption is considered as

$$t' \geq t + \Theta(q_s + q)T.$$

**VIII. PERFORMANCE ANALYSIS**

Here, we evaluate the efficiency of our CLS scheme and also compare it with the CLS schemes proposed by Karati et al. [9], Zhang et al. [20], Huang et al. [15], Choi et al. [17], He et al. [21], Tsai et al. [22], and Yuan et al. [23], in terms of computation time, security type and complexity assumption. We also present details about the experiment, efficiency and signature length of our CLS scheme.

**A. RESULTS OF EXPERIMENT**

To evaluate the efficiency of our CLS scheme, we preformed implementations with language C coding on a personal computer, Dell Intel(R) Core(TM) i5-8500 CPU @ 3.90-GHz processor, 16 GB RAM, Ubuntu 16.4 LTS, using GNU Multiple Precision Arithmetic (GMP) library and Pairing Based Cryptography (PBC) library. In Table 1, we show computation costs of our CLS scheme and the other CLS schemes.

In Table 1, the computation costs are measured according to the pre-computation costs of operations over bilinear pairings in PBC library. We selected Type-A pairings with 512-b group and embedding degree 2 equals to the 1024-b RSA security level. Type-A pairing is constructed on the super singular elliptic curve $y^2 = x^3 + x$ over the field $F_q$, where $|\mathcal{G}| = |\mathcal{G}_T| = |\mathcal{Z}_p|^{*}$ are prime ordered groups of points $E(F_q)$. In the next sub-section, we discuss computational time and signature length of our proposed CLS scheme in detail.

**B. COMPUTATIONAL TIME**

In Table 1, the time cost of signature generation and signature verification are shown according to the time cost of cryptographic primitive operations.

In our CLS scheme, the time cost of Setup algorithm consists of generating a pair of prime ordered group $(\mathcal{G}, \mathcal{G}_T)$ and one exponentiation. The cost of Set-Partial-Private-Key algorithm and Set-Public-Key algorithm is one exponentiation each. In addition, the signer requires to spend one exponentiation for running CLS-Sign algorithm. The CLS-Verify algorithm requires two exponentiations and one pairing computation in order that the verifier can check the validity of a signature.

The total computation cost of our proposed CLS scheme equals to 5.45ms (Table 1), which consists of computation costs of running Setup algorithm, Set-Partial-Private-Key algorithm and partial-private-key verification, Set-Secret-Value algorithm, Set-Public-Key algorithm, CLS-Sign algorithm and CLS-Verify algorithm.

As it is presented in Figure 2, the computation time of our CLS scheme is the lowest among other schemes presented in Table 1. The unit of y-axis in this figure is milliseconds (ms). Our scheme offers a very short signature size as it is shown in Table 1 and therefore is the most efficient one.

**C. SIGNATURE LENGTH**

The CLS-Sign algorithm in our proposed CLS scheme is two elements: $r \in \mathbb{Z}_p$ and $S \in \mathcal{G}$, which is shorter than that of the proposed CLS scheme by Karati et al. [9]. The CLS-Sign algorithm in [9] generates two tuples and the length of signature is $2|\mathcal{G}_q|$. In addition, the length of signature algorithms in Scheme 1 proposed by Huang et al. [15], the CLS scheme proposed by Choi et al. [17], He et al. [21] and Tsai et al. [22] are $|\mathcal{G}_q|$, which are also short and efficient. However, the Sign algorithm in the proposed CLS scheme in [20] and [23] are two tuples and three tuples respectively, which are less efficient than our CLS scheme.

**IX. CONCLUSION**

The emerging Industrial Internet of Things (IIoT) can ameliorate productivity, flexibility and save costs in different areas of industry, however it can pose new data security issues, particularly authentication. In this paper, we analysed the security of the certificateless signature scheme (CLS) for IIoT proposed in [9]. We showed that the scheme is universally forgeable under chosen message attacks. This can make the scheme insecure under their defined adversarial models of Type-I and Type-II. We proposed an entirely new
scheme that resists Type-I and Type-II adversaries. Our CLS scheme is the first lightweight scheme that can be proved with a very tight security reduction and is computationally efficient in comparison with other existing schemes. In order to show the computational efficiency of our proposed CLS scheme, we simulated our scheme and measured the time complexity. As a result, our CLS scheme is more secure and lightweight than other existing schemes and therefore more suitable for IoIOT applications.

REFERENCES


FATEMEH REZAEIBAGHA received her Bachelor of Information Technology Engineering from Azad University, Iran in 2009, her M.S. of Information Security from LTU, Sweden, in 2013 and her PhD in Computer Science from University of Wollongong, Australia in 2017. She is currently a lecturer in School of Engineering and Information Technology, Murdoch University, Perth, Australia. Her major research interests are cryptography, blockchain and cyber security.

KE HUANG received the M.S. degree in University of Electronic Science and Technology of China (UESTC). He is currently pursuing the Ph.D. degree in the College of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC). He was a visiting student with the University of Wollongong (UOW) from 2017 to 2019. His research interests are blockchain and Internet of Things (IoT).

YI MU received his PhD from Australian National University in 1994. He is currently a full professor at Fujian Normal University, China. Prior to his position at Fujian Normal University, he was a full professor at University of Wollongong (UOW), Australia. He also served as the Head of School of Computer Science and Software Engineering at UOW during 2011 and 2015. His research interests include cryptography and information security. He is a senior member of the IEEE.

XINYI HUANG received his Ph.D. degree from the School of Computer Science and Software Engineering, University of Wollongong, Australia, in 2009. He is currently a Professor at the Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Mathematics and Informatics, Fujian Normal University, China. His research interests include cryptography and information security. He has published over 100 research papers in refereed international conferences and journals, such as PKC, IEEE TOC, IEEE TPDS, IEEE TIFS, and IEEE Journal on Selected Areas in Communications. He is in the Editorial Board of IEEE TDSC and International Journal of Information Security. He has served as the program/general chair or program committee member in over 80 international conferences.

WENJIE YANG received his PhD degree from the College of Cyber Security College of Information Science and Technology, Jinan University, China, in 2018. He is currently a lecturer at the Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Mathematics and Informatics, Fujian Normal University, China. His research interests include cryptography and information security.