Hindawi
Security and Communication Networks
Volume 2017, Article ID 5704865, 8 pages
https://doi.org/10.1155/2017/5704865

Research Article

Strongly Unforgeable Certificateless Signature Resisting Attacks from Malicious-But-Passive KGC

Wenjie Yang, Jian Weng, Weiqi Luo, and Anjia Yang

College of Information Science and Technology/College of Cyber Security, Jinan University, Guangzhou 510632, China

Correspondence should be addressed to Jian Weng; cryptjweng@gmail.com and Weiqi Luo; lwq@jnu.edu.cn

Received 18 May 2017; Revised 10 August 2017; Accepted 6 September 2017; Published 20 November 2017

Academic Editor: Salvatore D'Antonio

In digital signature, strong unforgeability requires that an attacker cannot forge a new signature on any previously signed/new messages, which is attractive in both theory and practice. Recently, a strongly unforgeable certificateless signature (CLS) scheme without random oracles was presented. In this paper, we firstly show that the scheme fails to achieve strong unforgeability by forging a new signature on a previously signed message under its adversarial model. Then, we point out that the scheme is also vulnerable to the malicious-but-passive key generation center (MKGC) attacks. Finally, we propose an improved strongly unforgeable CLS scheme in the standard model. The improved scheme not only meets the requirement of strong unforgeability but also withstands the MKGC attacks. To the best of our knowledge, we are the first to prove a CLS scheme to be strongly unforgeable against the MKGC attacks without using random oracles.

1. Introduction

In 1984, Shamir brought in the notion of identity-based public key cryptography [1] to avert the heavy certificate management problem in traditional certificate-based public key systems. In this scenario, a fully trusted key generation center (KGC) takes care of deriving the secret key from an entity's public identity information (e.g., personal identity number) which is directly used as the corresponding public key. From then on, lots of efforts have been made in this area [2–4]. However, these identity-based schemes are subject to the key escrow problem; that is, the KGC owns any entity's secret key and, therefore, can do anything on behalf of the entity whereas not being detected.

In order to avoid this problem, Al-Riyami and Paterson introduced a new notion named certificateless public key cryptography (CL-PKC) and formalized its security model [5], in which the KGC is considered as a semitrusted third party and only produces a partial secret key for an entity using its own master secret key, and each entity independently picks an extra secret value for themselves. As a consequence, the entity's secret key consists of the partial secret key and the secret value, which make an attacker not do anything instead of the entity if the attacker just has one of the two parts. In this scenario [5], the attacks only knowing the entity's secret value are denoted by public key replacement (PKR) attacks while the others merely obtaining the master secret key are denoted by honest-but-curious key generation center (HKGC) attacks. Before long, Au et al. addressed a malicious-but-passive key generation center (MKGC) attacks [6], in which an attacker can set some trapdoors in the system parameters adaptively rather than only being told the master secret key like in HKGC attacks. So far, CL-PKC has received many attention under those adversarial models.

As an important branch in certificateless settings, many studies have focused on certificateless signature (CLS) schemes [7–13]. Nevertheless, most of these early CLS schemes are only provably secure in the random oracle model [14], whose security may not be able to remain when the random oracle is instantiated with a concrete hash function. To fill the gap, Liu et al. proposed the first CLS scheme secure in the standard model [15] and then some modified schemes were put forward in [16–23]. Nevertheless, all of them only were proven to be secure against PKR attacks and HKGC attacks. Although some of them like [19, 22, 23] have been indicated to be secure against MKGC attacks, the authors do not provide a formal security proof under MKGC attacks without random oracles. Therefore, it is still an unresolved problem to construct a concrete CLS scheme provably secure against MKGC attacks in the standard model.

Recently, Hung et al. gave a certificateless signature scheme (HT scheme, for short) [21] and discussed its security under PKR attacks and HKGC attacks. They claimed that no attacker can forge a valid signature on a message, even given some previous signatures on the message. In other words, they stated that their scheme is strongly unforgeable under their security model. In this paper, we firstly illustrate that the HT scheme does not meet strong unforgeability as they claimed, by forging a new valid signature on any previously signed message. Further, we show that the HT scheme suffers from MKGC attacks by giving a concrete attack. At last, we propose an improved certificateless signature scheme which not only is strongly unforgeable but also resists MKGC attacks in the standard model.

This paper is organized as follows. In Section 2, we give some preliminaries. In Section 3, we review the HT scheme and present our attacks. In Section 4, we describe two building blocks based on which we then construct an improved strongly secure certificateless signature scheme in the standard model. Finally, the conclusion is given in Section 5.

2. Preliminaries

2.1. Bilinear Pairings and Complexity Assumption. In this paper, we adopt the standard term about bilinear pairings described in [21]. Here, $(q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, g)$ is a concrete instance, where q denotes a big prime, $\mathbb{G}_1$ and $\mathbb{G}_2$ denote two q order cyclic groups, $\hat{e}$ denotes an admissible bilinear mapping $\mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, and g denotes a generator in $\mathbb{G}_1$.

Discrete Logarithm (DL) Assumption. Given $\langle \mathbb{G}_1, g, h \rangle$, the DL problem is to find the integer a such that $h = g^a$. If the DL problem cannot be solved in polynomial time, the DL assumption holds.

Computational Diffie-Hellman (CDH) Assumption. Given $\langle \mathbb{G}_1, g, g^a, g^b \rangle$, the CDH problem is to compute group element g^{ab}. If the CDH problem cannot be solved in polynomial time, the CDH assumption holds.

2.2. Collision-Resistant Hash Function (CRHF) Assumption. Let $\mathscr{H} = \{H_k\}$ be a keyed hash family of functions $H_k : \{0,1\}^* \to \{0,1\}^n$ indexed by $k \in \mathscr{K}$. We say that algorithm $\mathscr{A}$ has advantage ϵ in breaking the collision resistance of $\mathscr{H}$ if

$$\Pr[\mathscr{A}(k) = (m_0, m_1) : m_0 \neq m_1, H_k(m_0) = H_k(m_1)] > \epsilon, \quad (1)$$

where the probability is taken over the random choice of $k \in \mathscr{K}$ and the random bits of $\mathscr{A}$. Here, a hash family $\mathscr{H}$ is (t, ϵ)-collision-resistant if no t-time adversary has advantage at least ϵ in breaking the collision resistance of $\mathscr{H}$.

2.3. Framework and Security Model. The five algorithms below which can be run in polynomial time comprise a certificateless signature scheme:

(i) *Setup*: when receiving a security parameter ψ, the algorithm outputs the master secret key msk and system parameters sp. sp is available for all the other algorithms.

(ii) *PSKExt*: when receiving the master secret key msk and an entity e, the algorithm outputs the partial secret key psk_e and sends it over a secure channel to the entity.

(iii) *SetUKey*: when receiving an entity e, this algorithm outputs the secret key sk_e and public key pk_e. Note that sk_e is composed of sv_e and psk_e, where sv_e denotes the secret value and is picked by the entity itself independently.

(iv) *Sign*: when receiving an entity's secret key sk_e and a message m, this algorithm outputs a signature σ.

(v) *Verify*: when receiving a signature σ, a message m, and an entity public key pk_e, this algorithm outputs either "accept" or "reject" relying on the signature validity.

Like [11, 21], two types of adversaries are taken into account in the above CLS definition. The first is a public key replacement (PKR) attacker who knows the targeted entity secret value but cannot request the entity partial secret key. The other is an honest-but-curious KGC (HKGC) attacker that is allowed to obtain the master secret key but cannot know the targeted entity secret value. Here, the strong security of a CLS scheme is captured by means of Games 1 and 2 between a challenger $\mathscr{C}$ and an attacker $\mathscr{A} \in \{\mathscr{A}_1, \mathscr{A}_2\}$.

Game 1 (for PKR attacker)

(i) *Init*: given a security parameter ψ, $\mathscr{C}$ invokes *Setup* to produce the master secret key msk and system parameters sp. sp is given to $\mathscr{A}$ and msk is kept by itself.

(ii) *Queries*: in this phase, $\mathscr{A}$ runs the following queries adaptively:

$\mathscr{O}^{pk}(e)$: receiving an entity e, $\mathscr{C}$ invokes *SetUkey* to obtain the entity public key pk_e and returns it to $\mathscr{A}$.

$\mathscr{O}^{rep}(e, pk'_e)$: receiving a new entity public key pk'_e, $\mathscr{C}$ finds and updates the corresponding item for the entity e.

$\mathscr{O}^{psk}(e)$: receiving an entity e, $\mathscr{C}$ invokes *PSKExt* to obtain the entity partial secret key psk_e and returns it to $\mathscr{A}$.

$\mathscr{O}^{sk}(e)$: receiving an entity e, $\mathscr{C}$ invokes *SetUkey* to obtain the entity secret key sk_e and returns it to $\mathscr{A}$. Here, $\mathscr{C}$ returns $\perp$ if the entity e has already appeared in $\mathscr{O}^{rep}(e, pk'_e)$.

$\mathscr{O}^{sign}(e, m)$: receiving an entity e and a message m, $\mathscr{C}$ invokes *SetUkey* to obtain sk_e and then performs *Sign* to generate the signature σ for m under sk_e. At last, $\mathscr{C}$ returns it to $\mathscr{A}$.

(iii) *Forgery*: $\mathcal{A}$ returns (σ^*, e^*, m^*) and wins if the following conditions hold:

(1) σ^* is a valid signature of m^* on e^*.

(2) e^* is not requested in $\mathcal{O}^{psk}(e)$ and $\mathcal{O}^{sk}(e)$.

(3) (σ^*, e^*, m^*) is not an output in $\mathcal{O}^{sign}(e, m)$.

Game 2 (for HKGC attacker)

(i) *Setup*: given a security parameter ψ, $\mathcal{C}$ invokes *Setup* to produce the master secret key *msk* and system parameters *sp*. Both *msk* and *sp* are given to $\mathcal{A}$.

(ii) *Queries*: here, $\mathcal{A}$ may request any oracles defined in *Game 1*, except for $\mathcal{O}^{rep}(e, pk'_e)$ and $\mathcal{O}^{psk}(e)$, in an adaptive manner. Note that it is unreasonable to ask $\mathcal{C}$ to respond to the signing queries if pk_e has been replaced.

(iii) *Forgery*: $\mathcal{A}$ outputs (σ^*, e^*, m^*) and wins if the following conditions hold:

(1) σ^* is a valid signature of m^* on e^*.

(2) e^* is not requested in $\mathcal{O}^{rep}(e, pk'_e)$ and $\mathcal{O}^{sk}(e)$.

(3) (σ^*, e^*, m^*) is not an output in $\mathcal{O}^{sign}(e, m)$.

Au et al. illustrated the definition about malicious-but-passive KGC (MKGC) attacker, which is a stronger and more realistic security requirement. This MKGC may embed extra trapdoors in their system parameters before running *PSKExt*. To capture MKGC attacks, the following *Setup* algorithm is adopted in *Game 2*.

(i) *Setup*: $\mathcal{C}$ invokes $\mathcal{A}$ to initialize the system public/secret parameters (sp, msk). Here, to mount an attack more easily, $\mathcal{A}$ is allowed to set some trapdoors during the initialization phase.

Definition 1. If none of the adversaries can win the above two games with a nonnegligible advantage, in a probabilistic polynomial time (PPT), the signature scheme in certificateless settings is strongly unforgeable against adaptive chosen message attacks.

3. Analysis of HT Scheme

3.1. Review on HT Scheme. Here, we restate HT scheme [21] by the following five algorithms:

(i) *Setup*: given an instance $(q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, g)$, KGC firstly picks $\alpha \in \mathbb{Z}_p^*$ at random and computes $g_1 = g^\alpha \in \mathbb{G}_1$ and then sets the master secret key $msk = g_2^\alpha$, where $g_2 \in \mathbb{G}_1$. Furthermore, KGC randomly chooses $e_1, e_2, \ldots, e_{n_e}, e', s_1, s_2, \ldots, s_{n_v}, s', t_1, t_2, \ldots, t_{n_v}, t', w_1, w_2, \ldots, w_{n_w}, w' \in \mathbb{G}_1$ and sets four vectors $\overrightarrow{e} = \{e_i\}_{i=1}^{n_e}$, $\overrightarrow{s} = \{s_j\}_{j=1}^{n_v}$, $\overrightarrow{t} = \{t_j\}_{j=1}^{n_v}$, and $\overrightarrow{w} = \{w_k\}_{j=k}^{n_w}$, respectively. Next, KGC selects five classical collision-resistant hash functions $H_1 : \{0,1\}^* \rightarrow \{0,1\}^{n_e}$, $H_2, H_3 : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \{0,1\}^{n_v}$, $H_4 : \{0,1\}^* \rightarrow \{0,1\}^{n_w}$, and $H_5 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$, where n_e, n_v, and n_w are fixed lengths. Finally, KGC publishes $sp = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, g_1, g_2, e', \overrightarrow{e}, s', \overrightarrow{s}, t', \overrightarrow{t}, w', \overrightarrow{w}, H_1, H_2, H_3, H_4, H_5)$.

(ii) *PSKExt*: let $\overrightarrow{ve} = H_1(e) = (ve_1, ve_2, \ldots, ve_{n_e})$ be a bit string from hashing an entity e. Next, KGC randomly picks r_e from $\mathbb{Z}_p^*$ and computes $psk_e = (psk_{e1}, psk_{e2}) = (g_2^\alpha E^{r_e}, g^{r_e})$, where $E = e' \prod_{i=1}^{n_e} e_i^{ve_i}$. Finally, KGC secretly transmits psk_e to the entity.

(iii) *SetUkey*: given an entity e, this algorithm firstly chooses two random secret values $\theta_1, \theta_2 \in \mathbb{Z}_p^*$, generates the entity public key $pk_e = (pk_{e1}, pk_{e2}) = (g^{\theta_1}, g^{\theta_2})$, and then computes $\overrightarrow{vs} = H_2(pk_{e1}, pk_{e2}) = (vs_1, \ldots, vs_{n_v})$ and $\overrightarrow{vt} = H_3(pk_{e1}, pk_{e2}) = (vt_1, \ldots, vt_{n_v})$, where $\overrightarrow{vs}$ and $\overrightarrow{vt}$ are two bit strings of length n_v. Finally, the entity secret value is set to $sv_e = g_2^{\theta_1} S^{\theta_1} T^{\theta_2}$, where $S = s' \prod_{j=1}^{n_v} s_j^{vs_j}$ and $T = t' \prod_{j=1}^{n_v} t_j^{vt_j}$. Here, $sk_e = (sv_e, psk_e)$.

(iv) *Sign*: given a message m, an entity public key pk_e, and an entity secret key sk_e, this algorithm randomly selects $r_w \in \mathbb{Z}_p^*$, and computes $\overrightarrow{vw} = H_4(m) = (vw_1, vw_2, \ldots, vw_{n_w})$ and $h = H_5(m \parallel g^{r_w})$. Then, the signature σ is set by computing $(\sigma_1, \sigma_2, \sigma_3) = (psk_{e1}^h (sv_e)^h W^{r_w}, psk_{e2}^h, g^{r_w})$, where $W = (w' \prod_{k=1}^{n_w} w_k^{vw_k})$.

(v) *Verify*: given σ on m under e with pk_e, this algorithm can verify σ's validity by the following equation:

$$\hat{e}(\sigma_1, g) \stackrel{?}{=} \hat{e}(g_1, g_2)^h \hat{e}(\sigma_2, E) \hat{e}(pk_{e1}, g_2 S)^h \cdot \hat{e}(pk_{e2}, T)^h \hat{e}(\sigma_3, W), \quad (2)$$

where $h = H_5(m \parallel \sigma_3)$, $E = e' \prod_{i=1}^{n_e} e_i^{ve_i}$, $S = s' \prod_{j=1}^{n_v} s_j^{vs_j}$, $T = t' \prod_{j=1}^{n_v} t_j^{vt_j}$, and $W = (w' \prod_{k=1}^{n_w} w_k^{vw_k})$.

3.2. Attacks on HT Scheme

3.2.1. Strong Forgery Attack. Here, we indicate that an attacker $\mathcal{A}$ generates a new forgery σ' from an existing signature σ on m by interacting with the challenger $\mathcal{C}$.

Stage 1. $\mathcal{C}$ normally invokes the *Setup* algorithm to produce $msk = g_2^\alpha$ and $sp = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, g_1, g_2, e', \overrightarrow{e}, s', \overrightarrow{s}, t', \overrightarrow{t}, w', \overrightarrow{w}, H_1, H_2, H_3, H_4, H_5)$. Here, $\mathcal{C}$ sends *sp* to $\mathcal{A}$ and keeps *msk* by itself.

Stage 2. Given a target entity e and a message m, $\mathcal{C}$ first invokes the *PSKExt* and *SetUkey* algorithms to generate the entity secret key $(psk_{e1}, psk_{e2}, sv_e) = (g_2^\alpha E^{r_e}, g^{r_e}, g_2^{\theta_1} S^{\theta_1} T^{\theta_2})$ and then invokes the *Sign* algorithm to obtain a signature σ of m on e. Obviously, the signature concrete forms are $(\sigma_1, \sigma_2, \sigma_3) = (psk_{e1}^h (sv_e)^h W^{r_w}, psk_{e2}^h, g^{r_w})$, where $h = H_3(m \parallel g^{r_w})$.

Stage 3. In order to generate σ' for m under e, $\mathscr{A}$ randomly picks $r'_e \in \mathbb{Z}_p^*$ and sets $\sigma'_1 = \sigma_1 E^{r'_e h} = g_2^{\alpha h} E^{(r_e + r'_e)h} sv_e^h W^{r_w}$, $\sigma'_2 = \sigma_2 g^{r'_e h} = g^{(r_e + r'_e)h}$, and $\sigma'_3 = \sigma_3$.

It is clear that σ' is a new valid signature for m on e with pk_e since

$$\begin{aligned} \hat{e}\left(\sigma'_1, g\right) &= \hat{e}\left(g_2^{\alpha h} E^{(r_e+r'_e)h} sv_e^h W^{r_w}, g\right) = \hat{e}\left(g_2^{\alpha h}, g\right) \\ &\cdot \hat{e}\left(E^{(r_e+r'_e)h}, g\right) \hat{e}\left(\left(g_2 S\right)^{\theta_1 h} T_e^{\theta_2 h}, g\right) \hat{e}\left(W^{r_w}, g\right) \\ &= \hat{e}\left(g_1, g_2\right)^h \hat{e}\left(E, \sigma'_2\right) \hat{e}\left(pk_{e1}, g_2 S\right)^h \hat{e}\left(pk_{e2}, T\right)^h \\ &\cdot \hat{e}\left(W, \sigma'_3\right). \end{aligned} \tag{3}$$

Remark 2. We notice that there is a flaw in Hung et al.'s security proof. More specifically, when receiving a signing query on (m, e), $\mathscr{C}$ first picks $r_w \in \mathbb{Z}_p^*$ at random and then sets the hash value $h = H_5(m \parallel g^{r_w})$ to successfully compute the simulated signature $\sigma = (\sigma_1, \sigma_2, \sigma_3)$. According to Hung et al.'s simulation process, we know that $\sigma_3 = g^{r'_w}$, where $r'_w = r_w - \theta_1 h Q(\overrightarrow{vs})/K(\overrightarrow{vw})$, $r_w - (ah + \theta_1 h Q(\overrightarrow{vs}))/K(\overrightarrow{vw})$, or $r_w - ah/K(\overrightarrow{vw})$. Obviously, the verification equation $\hat{e}(\sigma_1, g) \overset{?}{=} \hat{e}(g_1, g_2)^{h'} \hat{e}(\sigma_2, E) \hat{e}(pk_{e1}, g_2 S)^{h'} \hat{e}(pk_{e2}, T)^{h'} \hat{e}(\sigma_3, W)$ is always not true since $h \neq h'$, where $h' = H_5(m \parallel \sigma_3)$. That is to say, $\mathscr{C}$ cannot always give a valid response to a signing query required by an attacker $\mathscr{A}$ with nonnegligible probability.

3.2.2. MKGC Attack. Here, we will show that the HT scheme cannot capture MKGC attacks by an interaction between $\mathscr{C}$ and $\mathscr{A}$ as follows.

Stage 1. Here, $\mathscr{A}$ is responsible for initializing the system parameters and sends them to $\mathscr{C}$. In particular, $\mathscr{A}$ adaptively sets the parameters $g_2 = g^{\beta}$, $s' = g^{x'}$, $s_1 = g^{x_1}, \ldots, s_n = g^{x_{n_v}}$, $t' = g^{y'}$, $t_1 = g^{y_1}, \ldots, t_n = g^{y_{n_v}}$, where $\beta, x', x_1, \ldots, x_{n_v}, y', y_1, \ldots, y_{n_v} \in \mathbb{Z}_p^*$.

Stage 2. In the queries phase, $\mathscr{A}$ can obtain the target entity public key $pk_e = (pk_{e1}, pk_{e2}) = (g^{\theta_1}, g^{\theta_2})$. Next, $\mathscr{A}$ derives the corresponding secret value $sv_e = g_2^{\theta_1} S^{\theta_1} T^{\theta_2}$ by computing $g_2^{\theta_1} = pk_{e1}^{\beta}$, $S^{\theta_1} = pk_{e1}^{x' \sum_{j=1}^{n_k} x_j^{vs_j}}$, and $T^{\theta_2} = pk_{e2}^{y' \sum_{j=1}^{n_k} y_j^{vt_j}}$, where $\overrightarrow{vs} = H_2(pk_{e1}, pk_{e2}) = (vs_1, \ldots, vs_{n_k})$ and $\overrightarrow{vt} = H_3(pk_{e1}, pk_{e2}) = (vt_1, \ldots, vt_{n_k})$.

Stage 3. Now, $\mathscr{A}$ can recover the entity full secret key (psk_e, sv_e) and sign any message m instead of the targeted entity.

Remark 3. In MKGC attacks, the KGC can adaptively set some trapdoor in the system parameters during the setup phase which may enhance its attack success probability. Obviously, in the security proof [21], the attacker passively received the master secret key and public parameters generated by the challenger like all the previous schemes [15–23], which did not consider how to capture the MKGC attacks.

4. Our CLS Scheme

In this section, we construct an improved strongly unforgeable certificateless signature scheme and demonstrate its security on the condition that both [3, 24] are secure, and DL and CRHF assumptions hold.

4.1. Building Blocks

Paterson and Schuldt's Identity-Based Signature Scheme (PIBS) [3]

(i) *Setup*: let $(q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, g)$ be an instance described in Section 2. First, pick a random element α from $\mathbb{Z}_p^*$ and compute $g_1 = g^{\alpha}$. Then, choose random elements g_2, e', w' from $\mathbb{G}_1$ and two vectors $\overrightarrow{e} = \{e_k\}_{k=1}^{n_e}$, $\overrightarrow{w} = \{w_l\}_{l=1}^{n_w}$ of lengths n_e and n_w, respectively, whose entries are random elements from $\mathbb{G}_1$. Finally, the master secret key is $msk = g_2^{\alpha}$ and the system parameter is $sp = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, g_1, g_2, e', \overrightarrow{e}, w', \overrightarrow{w})$.

(ii) *Extract*: let $E = H_e(e)$, where e is an entity. Set $\mathscr{E} = \{k \mid E[k] = 1, k = 1, 2, \ldots, n_e\}$. To construct e's secret key sk_e, pick a random number r_e from $\mathbb{Z}_p^*$ and compute

$$sk_e = (sk_{e1}, sk_{e2}) = \left(g_2^{\alpha_1}\left(e' \prod_{k \in \mathscr{E}} e_k\right)^{r_e}, g^{r_e}\right). \tag{4}$$

(iii) *Sign*: let $\mathscr{W} \subset \{1, 2, \ldots, n_m\}$ be the set of indices l such that $W[l] = 1$, where $W[l]$ is the lth bit of message m. σ is constructed by picking a random element r_w from $\mathbb{Z}_p^*$ and computing

$$\sigma = \left(g_2^{\alpha}\left(e' \prod_{k \in \mathscr{E}} e_k\right)^{r_e}\left(w' \prod_{l \in \mathscr{W}} w_l\right)^{r_m}, g^{r_e}, g^{r_w}\right). \tag{5}$$

(iv) *Verify*: given a signature $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ of m under e, a verifier accepts σ if the following equality holds:

$$\begin{aligned} &\hat{e}(\sigma_1, g) \\ &\overset{?}{=} \hat{e}(g_1, g_2)\, \hat{e}\left(e' \prod_{k \in \mathscr{E}} e_k, \sigma_2\right) \hat{e}\left(w' \prod_{l \in \mathscr{W}} w_l, \sigma_3\right). \end{aligned} \tag{6}$$

Output 1 if it is valid. Otherwise, output 0.

Boneh et al.'s Signature Scheme (BSW) [24]

(i) *SetUKey*: let $(q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, g)$ be an instance described in Section 2. First, pick a random number β from $\mathbb{Z}_p^*$ and set $g_{e1} = g^{\beta}$. Then, choose three random values g_{e2}, h_e, v'_e from $\mathbb{G}_1$ where $\log_g h_e$ is known and a random n_w-length vector $\overrightarrow{v}_e = \{v_{ei}\}_{i=1}^{n_w}$ whose elements are also chosen from $\mathbb{G}_1$. In addition, choose two random collision-resistant hash functions $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ and $H_w : \mathbb{G}_1 \rightarrow \{0,1\}^{n_e}$. Finally, output the secret key $sk_e = g_{e2}^{\beta}$ and the public key $pk_e = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, g, g_{e1}, g_{e2}, h_e, v_e', \overrightarrow{v}_e)$.

(ii) *Sign*: given a message m, first, pick two random numbers r_w, s from $\mathbb{Z}_p^*$ and compute $W = H_w(g^t h_e^s)$, where $t = H(m \parallel g^{r_w})$. Then, set $\mathscr{W} = \{i \mid W[i] = 1, i = 1, 2, \ldots, n_w\}$. At last, σ is constructed by computing

$$\sigma = (\sigma_1, \sigma_2, \sigma_3) = \left(g_{e2}^{\beta} \left(v_e' \prod_{i \in \mathscr{W}} v_{ei} \right)^{r_w}, g^{r_w}, s \right). \quad (7)$$

(iii) *Verify*: given σ of m under pk_e, first, compute $W = H_m(g^t h_e^{\sigma_3})$, where $t = H(m \parallel \sigma_2)$. Then, set $\mathscr{W} = \{i \mid N[i] = 1, i = 1, 2, \ldots, n_m\}$. Finally, verify the following equation:

$$\hat{e}(\sigma_1, g) \stackrel{?}{=} \hat{e}(g_{e1}, g_{e2})\, \hat{e}\left(v_e' \prod_{i \in \mathscr{W}} v_{ei}, \sigma_2 \right). \quad (8)$$

4.2. Our Concrete Scheme. Let $(q, \mathbb{G}_1, \mathbb{G}_2, \hat{e}, g)$ be an instance described in Section 2. In our scheme, three classic hash functions, $H : \{0,1\}^* \times sp \times pk_e \times e \times \mathbb{G}_1^2 \rightarrow \mathbb{Z}_p^*$, $H_w : \mathbb{G}_1 \rightarrow \{0,1\}^{n_w}$, and $H_e : \{0,1\}^* \rightarrow \{0,1\}^{n_e}$, are adopted from $\mathscr{H}$ to handle arbitrary messages and identities. Note that the cyclic groups and hash functions are publicly confirmed by all interested parties.

(i) *Setup*: first, choose random elements $\alpha_1, \alpha_2, \varphi, x'$, $x_1, x_2, \ldots, x_{n_e}, y', y_1, y_2, \ldots, y_{n_w}$ from $\mathbb{Z}_p^*$. Then, compute $g_1 = g^{\alpha_1}$, $g_2 = g^{\alpha_2}$, $h = g^{\varphi}$, $e' = g^{x'}$, $\overrightarrow{e} = \{e_k\}_{k=1}^{n_e} = \{g^{x_k}\}_{k=1}^{n_e}$, $w' = g^{y'}$, $\overrightarrow{w} = \{w_l\}_{l=1}^{n_w} = \{g^{y_l}\}_{l=1}^{n_w}$. Finally, the master secret key is $msk = (\alpha_1, \alpha_2, \varphi, x', x_1, x_2, \ldots, x_{n_e}, y', y_1, y_2, \ldots, y_{n_w})$ and system parameter is $sp = (\mathbb{G}_1, \mathbb{G}_2, g, g_1, g_2, h, e', \overrightarrow{e}, w', \overrightarrow{w})$.

(ii) *PSKExt*: let $E = H_e(e)$ where e is an entity. Set $\mathscr{E} = \{k \mid E[k] = 1, k = 1, 2, \ldots, n_e\}$. To construct e's partial secret key psk_e, pick a random number r_e from $\mathbb{Z}_p^*$ and compute

$$psk_e = (psk_{e1}, psk_{e2}) = \left(g_2^{\alpha_1} \left(e' \prod_{k \in \mathscr{E}} e_k \right)^{r_e}, g^{r_e} \right). \quad (9)$$

(iii) *SetUKey*: first, choose random elements β_{e1}, β_{e2}, $\phi_e, z_e', z_{e1}, z_{e2}, \ldots, z_{en_w}$ from $\mathbb{Z}_p^*$. Then, compute $g_{e1} = g^{\beta_{e1}}$, $g_{e2} = g^{\beta_{e2}}$, $h_e = g^{\phi_e}$, $v_e' = g^{z_e'}$, $\overrightarrow{v}_e = \{v_{ei}\}_{i=1}^{n_w} = \{g^{z_{ei}}\}_{i=1}^{n_w}$. At last, let the public key be $pk_e = (g_{e1}, g_{e2}, h_e, v_e', \overrightarrow{v}_e)$ and the secret values be $sv_e = (\beta_{e1}, \beta_{e2}, \phi_e, z_e', z_{e1}, z_{e2}, \ldots, z_{en_w})$. Note that $sk_e = (sv_e, psk_e)$.

(iv) *Sign*: given m, pk_e, and sk_e, the signer selects two random numbers $r_w, s \in \mathbb{Z}_p^*$, computes $t = H(m \parallel sp \parallel pk_e \parallel e \parallel psk_{e2} \parallel g^{r_w})$, and sets $\mathscr{E} = \{k \mid E[k] = 1, k = 1, 2, \ldots, n_e\}$, where $E = H_e(e)$, $\mathscr{W} = \{l \mid W[l] = 1, l = 1, 2, \ldots, n_w\}$, where $W = H_w(g^t h^s)$, and $\mathscr{W}' = \{i \mid W'[i] = 1, i = 1, 2, \ldots, n_w\}$, where $W' = H_w(g^t h_e^s)$, respectively. At last, σ is constructed by computing

$$\sigma = \left(g_{e2}^{\beta_{e1}} \left(v_e' \prod_{i \in \mathscr{W}'} v_{ei} \right)^{r_w}, g_2^{\alpha_1} \left(e' \prod_{k \in \mathscr{E}} e_k \right)^{r_e} \cdot \left(w' \prod_{l \in \mathscr{W}} w_l \right)^{r_w}, g^{r_e}, g^{r_w}, s \right). \quad (10)$$

(v) *Verify*: given σ of m under e, any verifier first computes $E = H_e(e)$, $W = H_w(g^t h^{\sigma_5})$ and $W' = H_w(g^t h_e^{\sigma_5})$, where $t = H(m \parallel sp \parallel pk_e \parallel e \parallel \sigma_3 \parallel \sigma_4)$. Then, the verifier sets $\mathscr{E} = \{k \mid E[k] = 1, k = 1, 2, \ldots, n_e\}$, $\mathscr{W} = \{l \mid W[l] = 1, l = 1, 2, \ldots, n_w\}$, and $\mathscr{W}' = \{i \mid W'[i] = 1, i = 1, 2, \ldots, n_w\}$, respectively. Finally, the equalities are checked as follows:

$$\begin{aligned} \hat{e}(\sigma_1, g) &\stackrel{?}{=} \hat{e}(g_{e1}, g_{e2})\, \hat{e}\left(v_e' \prod_{i \in \mathscr{W}'} v_{ei}, \sigma_4 \right), \\ \hat{e}(\sigma_2, g) & \\ &\stackrel{?}{=} \hat{e}(g_1, g_2)\, \hat{e}\left(e' \prod_{k \in \mathscr{E}} e_k, \sigma_3 \right) \hat{e}\left(w' \prod_{l \in \mathscr{W}} w_l, \sigma_4 \right). \end{aligned} \quad (11)$$

Output 1 if the equations hold. Otherwise, output 0.

For simplicity, we may set $msk = \alpha_1$ and $sv_e = \beta_{e1}$ because the others called implicit master secret keys/secret values are not used during its execution. Here, they are explicitly listed to easily prove its security as follows.

4.3. Security Analysis. In this subsection, we introduce two lemmas to demonstrate that our scheme is strongly unforgeable against PKR attacks and MKGC attacks based on the CDH, DL and CRHF assumptions defined in Section 2.

Lemma 4. *Our scheme is strongly secure against PKR attacks launched by the attacker $\mathscr{A}_1$ assuming PIBS is weakly secure [3], the discrete log problem is intractable in $\mathbb{G}_1$, and finding concrete collisions is difficult in $\mathscr{H}$.*

Analysis. Given any PPT adversary $\mathscr{A}_1$ trying to break our CLS scheme in an adaptive chosen-message attack, we can forge a PPT adversary $\mathscr{B}_1$ producing a weak forgery on the *PIBS* scheme/solving discrete log in $\mathbb{G}_1$/finding concrete collision of $\mathscr{H}$. Moreover, to consistently answer $\mathscr{A}_1$'s queries, $\mathscr{B}_1$ keeps a table T which is initially empty in our security proofs.

Init. First, if $\mathscr{B}_1$ attempt to break the *PIBS* scheme, $\mathscr{B}_1$ randomly picks a number φ from $\mathbb{Z}_p^*$ and sets $h = g^{\varphi}$, else $\mathscr{B}_1$ directly picks a random value h from $\mathbb{G}_1$. Then, $\mathscr{B}_1$ chooses three hash functions H, H_e, and H_w from a classic hash family $\mathscr{H}$. Finally, $\mathscr{B}_1$ returns (PIBS.*params*, h, H, H_e, H_w) to $\mathscr{A}_1$ as the system parameters sp. Here, in order to make the initialization appropriate, we flip a coin to predict the forgery type launched by $\mathscr{A}_1$.

Queries. In the query phase, $\mathscr{B}_1$ responds to $\mathscr{A}_1$'s all queries defined in their security model as follows:

$\mathscr{O}^{psk}(e)$: $\mathscr{B}_1$ runs *PIBS.Extract* on $H_e(e)$ to obtain the partial secret key psk_e and returns it to $\mathscr{A}_1$.

$\mathscr{O}^{sk}(e)$: $\mathscr{B}_1$ searches T to find the entity e and get the entity secret value $sv_e = (\beta_{e1}, \beta_{e2}, \phi_e, z'_e, z_{e1}, z_{e2}, \ldots, z_{en_w})$. If it does not exist, $\mathscr{B}_1$ first picks $n_w + 4$ random values $(\beta_{e1}, \beta_{e2}, \phi_e, z'_e, z_{e1}, z_{e2}, \ldots, z_{en_w})$ from $\mathbb{Z}_p^*$ and stores them in T. Finally, $\mathscr{B}_1$ returns (sv_e, psk_e) to $\mathscr{A}_1$ as the secret key for e.

$\mathscr{O}^{pk}(e)$: $\mathscr{B}_1$ searches T to discover the entity e and returns $(g_{e1}, g_{e2}, v'_e, v_{e1}, \ldots, v_{en_w})$ to $\mathscr{A}_1$. Otherwise, $\mathscr{B}_1$ first picks $n_w + 4$ random values $(\beta_{e1}, \beta_{e2}, \phi_e, z'_e, z_{e1}, z_{e2}, \ldots, z_{en_w})$ from $\mathbb{Z}_p^*$ and then stores those values in L. At last, $(g^{\beta_{e1}}, g^{\beta_{e2}}, \phi_e, g^{z'_e}, g^{z_{e1}}, \ldots, g^{z_{en_w}})$ is returned to $\mathscr{A}_1$ as the entity public key.

$\mathscr{O}^{rep}(e, pk'_e)$: $\mathscr{B}_1$ searches T to find the entity e and update its public key using a new public key pk'_e. If it does not exist, $\mathscr{A}_1$ directly sets the entity public key to be pk'_e.

$\mathscr{O}^{sign}(e, m)$: for a signature of a message m on the entity public key pk_e,

(i) if $\mathscr{B}_1$ attempts to break *PIBS* by means of $\mathscr{A}_1$'s ability, $\mathscr{B}_1$ picks a random value λ from $\mathbb{Z}_p$ and runs *PIBS.Sign*$(H_e(e), H_m(g^\lambda))$ to produce $(\sigma'_1, \sigma'_2, \sigma'_3)$. Moreover, $\mathscr{B}_1$ computes $t = H(m \parallel sp \parallel pk_e \parallel e \parallel \sigma'_2 \parallel \sigma'_3)$ to recover $s = (\lambda - t)/\varphi$.

(ii) Otherwise, $\mathscr{B}_1$ invokes *PPKExt*$(H_e(e))$ to generate e's partial secret key (psk_{e1}, psk_{e2}). Then, $\mathscr{B}_1$ picks two random values r_w, s from $\mathbb{Z}_p^*$ and simulates *PIBS.Sign*$(H_e(e), H_w(g^t h^s))$ to produce $(\sigma'_1, \sigma'_2, \sigma'_3)$ where $t = H(m \parallel sp \parallel pk_e \parallel e \parallel \sigma'_2 \parallel \sigma'_3)$.

Next, $\mathscr{B}_1$ computes $W' = H_w(g^t h_e^s)$ and sets $\sigma_1 = g^{\beta_{e1}\beta_{e2}} (\sigma'_3)^{\sum_{i \in \mathscr{W}'} z_{ei}}$ where $\mathscr{W}' = \{i \mid W'[i] = 1, i = 1, 2, \ldots, n_w\}$. At last, $\mathscr{B}_1$ sets $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5) = (\sigma_1, \sigma'_1, \sigma'_2, \sigma'_3, s)$ and returns it to $\mathscr{A}_1$.

Forgery. If $\mathscr{A}_1$ outputs σ^* on m^* for e^* with pk_e^* such that

(i) $H_w(g^{t^*} h^{s^*}) \neq H_w(g^t h^s)$, $\mathscr{B}_1$ can succeed in breaking the *PIBS* scheme by forging a valid signature $(\sigma_2^*, \sigma_3^*, \sigma_4^*)$ on the message $H_w(g^{t^*} h^{s^*})$;

(ii) $H_w(g^{t^*} h^{s^*}) = H_w(g^t h^s)$ but $g^{t^*} h^{s^*} \neq g^t h^s$, $\mathscr{B}_1$ can succeed in finding a concrete collision to hash function H_w;

(iii) $g^{t^*} h^{s^*} = g^t h^s$ but $t^* \neq t$, $\mathscr{B}_1$ can succeed in solving the discrete logarithm problem by computing $d = (t - t^*)/(s^* - s)$;

(iv) $g^{t^*} h^{s^*} = g^t h^s$, $t^* = t$ but $m^* \parallel sp \parallel pk_e^* \parallel e^* \parallel \sigma_3^* \parallel \sigma_4^* \neq m \parallel sp \parallel pk_e \parallel e \parallel \sigma_3 \parallel \sigma_4$, $\mathscr{B}_1$ can succeed in finding a concrete collision to hash function H.

Here, s, t, and $m \parallel sp \parallel pk_e \parallel e \parallel \sigma_3 \parallel \sigma_4$ denote those elements queried/involved during the query phase.

Lemma 5. *Our scheme is strongly secure against MKGC attacks launched by $\mathscr{A}_2$ assuming BSW is strongly existential unforgeable.*

Analysis. Given a PPT adversary $\mathscr{A}_2$ breaking our CLS scheme in an adaptive chosen-message attack, we simulate a PPT adversary $\mathscr{B}_2$ producing a strong forgery on the *BSW* scheme. Moreover, to consistently answer $\mathscr{A}_2$'s queries, $\mathscr{B}_2$ keeps a table T which is initially empty in our security proofs.

Init. $\mathscr{B}_2$ invokes $\mathscr{A}_2$ to initialize the master secret key $msk = (\alpha_1, \alpha_2, \varphi, x', x_1, x_2, \ldots, x_{n_e}, y', y_1, y_2, \ldots, y_{n_w})$ and the system parameters $sp = (\mathbb{G}_1, \mathbb{G}_2, g, h, g_1, g_2, e', \overrightarrow{e}, w', \overrightarrow{w})$. Here, in order to launch MKGC attacks more easily, $\mathscr{A}_2$ is allowed to set some trapdoors during the initialization phase.

Queries. In the query phase, $\mathscr{B}_2$ responds to $\mathscr{A}_2$'s all queries involved in their security model as follows:

$\mathscr{O}^{sk}(e)$: $\mathscr{B}_2$ searches T to find the entity e and get the entity secret value $sv_e = (\beta_{e1}, \beta_{e2}, \phi_e, z'_e, z_{e1}, z_{e2}, \ldots, z_{en_w})$. If it does not exist, $\mathscr{B}_2$ first picks $n_w + 4$ random values $(\beta_{e1}, \beta_{e2}, \phi_e, z'_e, z_{e1}, z_{e2}, \ldots, z_{en_w})$ from $\mathbb{Z}_p^*$ and stores them in T. Finally, $\mathscr{B}_2$ returns (sv_e, psk_e) to $\mathscr{A}_2$ as the secret key for e.

$\mathscr{O}^{pk}(e)$: Analogously, $\mathscr{B}_2$ looks up T to find the entity e and returns $(g_{e1}, g_{e2}, v'_e, v_{e1}, \ldots, v_{en_w})$ to $\mathscr{A}_2$. If e is not found, $\mathscr{B}_2$ first picks $n_w + 4$ random elements $(\beta_{e1}, \beta_{e2}, \phi_e, z'_e, z_{e1}, z_{e2}, \ldots, z_{en_w})$ from $\mathbb{Z}_p^*$ and then stores those values in T. At last, $\mathscr{B}_2$ sets $g_{e1} = g^{\beta_{e1}}$, $g_{e2} = g^{\beta_{e2}}$, $h_e = \phi_e$, $v'_e = g^{z'_e}$, $v_{e1} = g^{z_{e1}}, \ldots, v_{en_w} = g^{z_{en_w}}$ and returns these elements to $\mathscr{A}_2$.

$\mathscr{O}^{rep}(e, pk'_e)$: $\mathscr{B}_2$ searches T to find the entity e and updates the entity public key using a new public key pk'_e. If it does not exist, $\mathscr{B}_2$ directly sets the entity public key to be pk'_e provided by $\mathscr{A}_2$.

$\mathscr{O}^{sign}(e, m)$: for a signature query on a message m under an entity e with the public key pk_e, $\mathscr{B}_2$ first invokes *PPKExt*$(H_e(e))$ to get the entity partial secret key psk_e and then runs *BSW.Sign*$(pk_e, m \parallel sp \parallel pk_e \parallel e \parallel psk_{e2})$ to generate $(\sigma'_1, \sigma'_2, \sigma'_3)$. Next, $\mathscr{B}_2$ computes $t = H(m \parallel sp \parallel pk_e \parallel e \parallel psk_{e2} \parallel \sigma'_2)$ and sets $\sigma_2 = psk_{e1}(\sigma'_2)^{\sum_{l \in \mathscr{W}} y_l}$, where $\mathscr{W} = \{l \mid W[l] = 1, l = 1, 2, \ldots, n_w\}$ and $W = H_w(g^t h^{\sigma'_3})$. At last, $\mathscr{B}_2$ sets $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5) = (\sigma'_1, \sigma_2, D_2, \sigma'_2, \sigma'_3)$ and returns it to $\mathscr{A}_2$.

Forgery. If $\mathscr{A}_2$ eventually returns $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*, \sigma_5^*)$ for m^* under e^* with pk_e^*, $\mathscr{B}_1$ can succeed in breaking the *BSW* scheme by forging a valid signature $(\sigma_1^*, \sigma_4^*, \sigma_5^*)$ on the message $H_e(g^{t^*} h_e^{s^*})$ where $t^* = H(m^* \parallel sp \parallel pk_e^* \parallel e^* \parallel \sigma_3^* \parallel$

σ_4^*). Note that H and H_e are two types of classic hash in the BSW scheme.

In general, we demonstrate that our CLS scheme is strongly unforgeable against PKR attacks and MKGC attacks in the standard model by combining Lemmas 4 and 5.

4.4. Discussion. Although our scheme has more parameters compared with the existing schemes, it not only overcomes the weakness of those previous schemes but also is proven to be secure under our stronger security model. In Lemma 5, the KGC can set some trapdoors in the system parameters adaptively rather than passively receiving the master secret key and the corresponding public parameters from the challenger like in the previous schemes, which is consistent with the actual situation in a concrete scheme. To the best of our knowledge, we are the first to prove a CLS scheme to be strongly secure against the MKGC attacks without random oracles.

5. Conclusion

In this paper, we firstly showed that Hung et al.'s scheme cannot meet the requirements of strong unforgeability as they claimed. Then, we pointed out that their construction does not withstand the MKGC attacks and gave a concrete forgery to break it. Finally, we constructed an improved strongly unforgeable certificateless signature scheme and proved its security under the MKGC attacks in the standard model.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by National Science Foundation of China (Grants nos. 61373158, 61472165, 61732021, and 61702222), Guangdong Provincial Engineering Technology Research Center on Network Security Detection and Defence (Grant no. 2014B090904067), Guangdong Provincial Special Funds for Applied Technology Research and Development and Transformation of Important Scientific and Technological Achieve (Grant no. 2016B010124009), Guangzhou Key Laboratory of Data Security and Privacy Preserving, and Guangdong Key Laboratory of Data Security and Privacy Preserving.

References

[1] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proceedings of the 4th International Conference on the Theory and Application of Cryptographic Techniques (CRYPTO '84)*, pp. 47–53, Santa Barbara, Calif, USA, 1987.

[2] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Advances in Cryptology—CRYPTO 2001*, vol. 2139 of *Lecture Notes in Computer Science*, pp. 213–229, 2001.

[3] K. G. Paterson and J. C. N. Schuldt, "Efficient identity-based signatures secure in the standard model," in *Information Security and Privacy*, vol. 4058 of *Lecture Notes in Computer Science*, pp. 207–222, 2006.

[4] C. Sato, T. Okamoto, and E. Okamoto, "Strongly unforgeable ID-based signatures without random oracles," *International Journal of Applied Cryptography*, vol. 2, no. 1, pp. 35–45, 2010.

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

[6] M. H. Au, J. Chen, J. K. Liu, Y. Mu, D. S. Wong, and G. Yang, "Malicious KGC attacks in certificateless cryptography," in *Proceedings of the 2nd ACM Symposium on Information, Computer and Communications Security (ASIACCS '07)*, pp. 302–311, Singapore, Singapore, March 2007.

[7] D. H. Yum and P. J. Lee, "Generic construction of certificateless signature," in *Information Security and Privacy*, vol. 3108 of *Lecture Notes in Computer Science*, pp. 200–211, 2004.

[8] W. Yap, S. Heng, and B. Goi, "An Efficient Certificateless Signature Scheme," in *Emerging Directions in Embedded and Ubiquitous Computing*, vol. 4097 of *Lecture Notes in Computer Science*, pp. 322–331, 2006.

[9] Z. Zhang, D. S. Wong, J. Xu, and D. Feng, "Certificateless public-key signature: security model and efficient construction," in *Applied Cryptography and Network Security*, vol. 3989 of *Lecture Notes in Computer Science*, pp. 293–308, 2006.

[10] W. Yap, S. S. M. Chow, S. Heng, and B. Goi, "Security mediated certificateless signatures," in *Proceedings of the 5th International Conference on Applied Cryptography and Network Security (ACNS '07)*, pp. 459–477, Zhuhai, China.

[11] X. Huang, Y. Mu, W. Susilo, D. S. Wong, and W. Wu, "Certificateless signature revisited," in *Proceedings of the 12th Australasian Conference on Information Security and Privacy (ACISP '07)*, pp. 308–322, Townsville, Australia, 2007.

[12] S. Chang, D. S. Wong, Y. Mu, and Z. Zhang, "Certificateless threshold ring signature," *Information Sciences*, vol. 179, no. 20, pp. 3685–3696, 2009.

[13] Z. Wan, J. Weng, and J. Li, "Security mediated certificateless signatures without pairing," *Journal of Computers*, vol. 5, no. 12, pp. 1862–1869, 2010.

[14] R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology, revisited," *Journal of the ACM*, vol. 51, no. 4, pp. 557–594, 2004.

[15] J. K. Liu, M. H. Au, and W. Susilo, "Self-generated-certificate public key cryptography and certificateless signature/encryption scheme in the standard model," *ACR Cryptology ePrint Archive*, vol. 2006, article 373, 2006.

[16] H. Xiong, Z. Qin, and F. Li, "An improved certificateless signature scheme secure in the standard model," *Fundamenta Informaticae*, vol. 88, no. 1-2, pp. 193–206, 2008.

[17] Y. Yuan, D. Li, L. Tian, and H. Zhu, "Certificateless signature scheme without random oracles," in *Proceedings of the 3rd International Conference and Workshops on Advances in Information Security and Assurance (ISA '09)*, pp. 31–40, 2009.

[18] Y. Yu, Y. Mu, G. Wang, Q. Xia, and B. Yang, "Improved certificateless signature scheme provably secure in the standard model," *IET Information Security*, vol. 6, no. 2, pp. 102–110, 2012.

[19] Y. Yuan and C. Wang, "Certificateless signature scheme with security enhanced in the standard model," *Information Processing Letters*, vol. 114, no. 9, pp. 492–499, 2014.

[20] L. Cheng and Q. Wen, "Provably secure and efficient certificateless signatur in the standard model," *International Journal*

of Information and Communication Technology, vol. 7, no. 2-3, pp. 287–301, 2015.

[21] Y.-H. Hung, S.-S. Huang, Y.-M. Tseng, and T.-T. Tsai, "Certificateless signature with strong unforgeability in the standard model," *Informatica*, vol. 26, no. 4, pp. 663–684, 2016.

[22] L. Pang, Y. Hu, Y. Liu, K. Xu, and H. Li, "Efficient and secure certificateless signature scheme in the standard model," *International Journal of Communication Systems*, vol. 30, no. 5, Article ID e3041, 2017.

[23] S. Canard and V. C. Trinh, "An Efficient Certificateless Signature Scheme in the Standard Model," in *The Journal of Information System Security*, vol. 10063 of *Lecture Notes in Computer Science*, pp. 175–192, 2016.

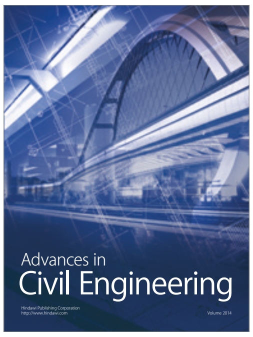
[24] D. Boneh, E. Shen, and B. Waters, "Strongly unforgeable signatures based on computational Diffie-Hellman," in *Public Key Cryptography—PKC 2006*, vol. 3958 of *Lecture Notes in Computer Science*, pp. 229–240, Springer, Berlin, Germany, 2006.

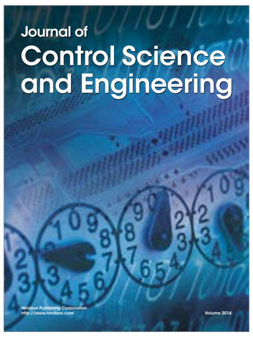
Journal of
Engineering

The Scientific World Journal

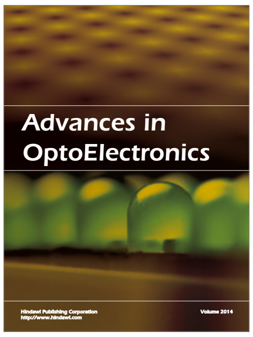
International Journal of
Rotating Machinery

Journal of
Sensors

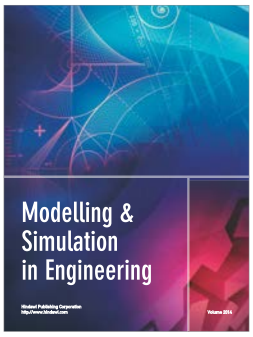
International Journal of
Distributed Sensor Networks

Advances in
Civil Engineering

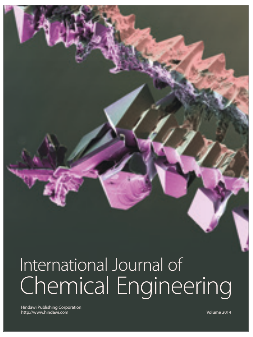
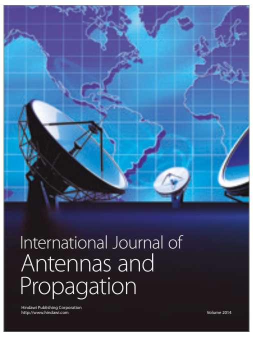

Journal of
Control Science and Engineering

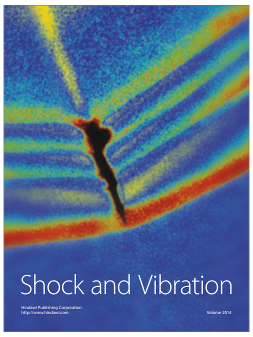
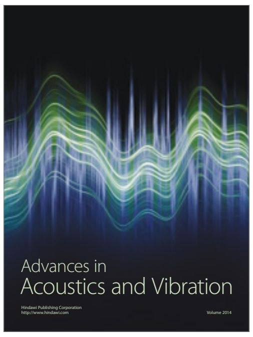
Journal of
Robotics

Journal of
Electrical and Computer Engineering

Advances in
OptoElectronics

VLSI Design

International Journal of
Navigation and Observation

Modelling & Simulation in Engineering

International Journal of
Aerospace Engineering

International Journal of
Chemical Engineering

International Journal of
Antennas and Propagation

Active and Passive Electronic Components

Shock and Vibration

Advances in
Acoustics and Vibration