Vol.2018-DPS-174 No.21
Vol.2018-CSEC-80 No.21
2018/3/6

A Certificateless Signature Scheme to Reduce Loads on Key Generation Center

Ei Khaing Win[1,a)] Yuuichi Teranishi[1,2] Yoshimasa Ishi[1] Tomoya Kawakami[3,1]
Tomoki Yoshihisa[1] Shinji Shimojo[1]

Abstract: Certificateless public key cryptography (CLPKC) eliminates the certificate and avoids key escrow problem. In CLPKC, Key Generation Center (KGC) is responsible for generating partial private key and revoking a user is an important problem. There have been some solutions to revoke users in CLPKC. Though an existing solution proposes a pairing-free revocation scheme, it still requires high time key computation cost of exponentiation operations on KGC. In this paper, we propose a revocable certificate-less signature scheme which requires less computation cost than existing schemes. In the proposed scheme, valid users receive the time key, which is calculated using only one exponentiation operation and some hash operations on KGC for a revocation. Under the discrete logarithm problem assumption, we prove that the proposed scheme is secure against existential forgeability in the random oracles.

1. Introduction

In traditional public key cryptography (PKC) [1], each user generates a pair of keys; public key and private key. To prove the valid users, the certificate authority issues a digital certificate that associates public key with the user's identity. Because PKC uses certificates for authentication, public key cryptography introduces the drawbacks such as the certificate management overhead for revocation, verification, and large storage. To simplify the certificate overhead, Shamir proposed identity-based cryptography (IBC) in which unique strings are used to represent an individual or organization [2]. Although identity-based encryption eliminates certificates, a trusted third party called the private key generator (PKG) exists. Because PKG generates private keys for all users, it can decrypt all exchanged messages. That is called the key escrow problem. In 2003, Al Riyami and Patterson proposed certificateless public key cryptography (CLPKC) to resolve the key escrow problem [3]. In CLPKC, key generation center (KGC) generates partial private keys as proof of validity. In addition, the users have to create public and private keys. Since the users generate own key pair, unlike PKG in IBC, KGC has no way to know the private keys to decrypt the message. Therefore, certificateless public key cryptography is a solution to prove valid users for the systems where the users control data access themselves.

In some cases, the validity status of a user should be changed. In other words, user revocation is necessary whenever there is compromised, illegal or misbehaved user. In PKC, certificate authority prepares certificate revocation lists (CRLs) so that the users query the lists to know the revoked users. For efficient certificate revocation, some studies have been done [4], [5]. To solve the revocation problem, PKG produces new time keys periodically and send them to non-revoked users via secure channels [6] or PKG uses some data structure to achieve efficiency [7] in identity-based cryptography. Instead of expensive secure channels, some identity-based schemes that require public channels have been proposed [8]. Regarding user revocation, there are existing solutions based on KGC or online mediator called the security mediator (SEM) in certificateless public key encryption ([18]-[25]).

To achieve secure data delivery in IoT applications, the authentication of the sender or source authentication is one of the necessary security features to be provided. Besides, the data owners do not place total trust on third party. In other words, although third party exists to prove user validity, users want to prevent sensitive data from third party. Moreover, it is important to revoke the receiver who misbehaves or whose keys are compromised. Therefore, revocable certificateless signature scheme is suitable for IoT applications. Although the SEM helps reduce the load of KGC for revocation, it is a point of the security threats [18]. Eliminating the SEM, some works introduce SEM free signature schemes. In SEM free revocable CLS schemes ([21]-[25]), new time key generation is a burden for the KGC as the key-update cost increases logarithmically in the number of non-revoked users. Moreover, in IoT applications, efficiency is a desirable property as the users and KGC may be mobile devices with limited storage and computing power.

Contributions. In this paper, we propose an efficient revocable certificateless signature scheme, which reduces the computational cost of KGC for user revocation. Likewise existing SEM free revocable certificateless signature schemes without pairings, our proposed scheme avoids pairing operations and uses the public channel to transmit new time keys to non-revoked users. We reduced computation for new time key generation by reducing the number of modular exponentiation.

1 Osaka University, Suita, Osaka, Japan
2 NICT, Koganei, Tokyo, Japan
3 NAIST, Ikoma, Nara, Japan
a) ei.khaing.win@ais.cmc.osaka-u.ac.jp

2. Related Work

To avoid the certificate overhead and key escrow problem, Al Riyami and Paterson proposed certificateless public key cryptography (CLPKC) in 2003 [3]. In 2005, [9] constructed an efficient certificateless public key encryption was constructed. And many certificateless encryption schemes ([10]-[13]) and certificateless signature schemes ([14]-[16]) have been presented. To enhance the performance, pairing free certificateless signature scheme was presented without considering revocation [17]. Applying on-line mediator called the Security Mediator (SEM), efficient revocation scheme was proposed for certificate public key encryption [18]. The idea is that KGC splits a partial private key into two secret keys. Then, it transmits one secret key to user and another secret key to the SEM via secure channels. The SEM helps only non-revoked users for every decryption. By stopping help, the SEM handles user revocation. Based on the same idea, some previous works have been proposed as the revocation schemes [19], [20]. Eliminating the SEM, in SEM free schemes, KGC performs new time key generation in some proposed papers [21], [22], [23]. In 2014, a revocable certificateless signature scheme that transmits the time keys via public channels was proposed in [22]. To enhance the efficiency for signing and verification of the signature scheme [22], [23] proposed a strongly unforgeable revocable certificateless signature scheme. The scheme reduces one scalar multiplication in the signing and two pairing operations in the verification. Pairing free revocable certificateless signature scheme was first proposed by [24]. In [24], new time key distribution is done via public channels. And [25] presented a revocable certificateless signature scheme by adding security proof for the scheme [24]. However, in every time key update of KGC, there is one exponentiation requirement for each non-revoked user. And the number of exponentiation operations grows linearly with the number of non-revoked users and time key update.

3. Definitions

3.1 Revocable Certificateless Signature

A revocable certificateless signature scheme (RCLS) consists of eight algorithms.

- Setup: The KGC runs this algorithm to generate public parameters $params$ and master secret key m_k.
- Extract-Partial-Private-Key: This algorithm takes $params$, m_k and an identity ID as input and generates a partial private key D_{ID}. The KGC runs this algorithm and transmits the partial private key to the user via secure channel.
- Update-time-Key: Firstly, this algorithm generates W_t for all non-revoked identities at time t just for once. Then, using $params$, x_{ID}, D_{ID}, m_k and an identity ID as input, the KGC runs this algorithm to generate the periodic key $D_{ID,t}$. Then, $D_{ID,t}$ is transmitted to the user via public channel.
- Set-Secret-Value: The user with ID runs this algorithm to produce a secret value s_{ID}.
- Set-Private-Key: Taking $params$, s_{ID}, D_{ID}, and $D_{ID,t}$ as input, the user runs this algorithm. And the algorithm outputs the private key $SK_{ID,t}$.
- Set-Public-Key: The user with ID runs this algorithm by taking $params$ and s_{ID}. The output is public key PK_{ID}.
- Sign: The sender with ID runs this algorithm to sign the message M. The algorithm takes $params$, ID, time t, $SK_{ID,t}$ and M as inputs and outputs the signature σ.
- Verify: The receivers run this algorithm to verify the signature. The algorithm takes $params$, PK_{ID}, X_{ID}, t, M, W_t and signature σ as input. After verifying the signature, the algorithm outputs "accept" or "reject".

3.2 Security Model

We consider three types of adversaries; Type-I, Type-II and Type-III adversaries to prove the security of the proposed scheme.

- Type-I adversary: Adversary who does not possess the master secret key. The adversary is allowed to replace the public key.
- Type-II adversary: An honest but curious KGC. Therefore, adversary possesses the master secret key although public key replacement is not allowed.
- Type-III adversary: A revoked user. The adversary who has partial private key but does not have new time key.

A revocable certificateless signature scheme is existentially unforgeable against chosen-ciphertext attacks if there is no polynomial-time adversary with non-negligible advantage. For security proofs of the signature scheme, we define three security games, each corresponding to one type of adversaries described above. For all types of adversaries, if the adversary can output a valid signature, then the adversary wins the game.

3.3 Security Game for Type-I adversary

The challenger runs setup algorithm to generate public parameters and master secret key. Then, the parameters are given to adversary $\mathcal{A}$. Master secret key is kept secret.

- Queries: Type-I adversary can query the following oracles.
  - Partial-Private-Key-Extraction: The challenger runs the Extract-Partial-Private-Key algorithm to generate partial private key. Then the partial private key is given to $\mathcal{A}$.
  - Time-Key-Query: The challenger runs the Update-time-Key algorithm to generate the new time key. Then the challenger returns it to the adversary $\mathcal{A}$.
  - Secret-Value-Query: The challenger runs Set-Secret-Value algorithm to generate the secret value, then returns it to the adversary $\mathcal{A}$.
  - Public-Key-Request: For the adversary's query for public key, the challenger runs Set-Public-Key algorithm to generate public key for the adversary $\mathcal{A}$.
  - Public-Key-Replace: This query allows the adversary $\mathcal{A}$ to replace the public key with any different value.
  - Signature: The challenger runs Sign algorithm to generate the signature for the tuple of (M, ID, t).
- Forge: The adversary outputs a signature on the tuple (M^*, ID^*, T^*). And the tuple (M^*, ID^*, T^*) is the one that is never used in the Signature query.

3.4 Security Game for Type-II adversary

The challenger runs setup algorithm to generate public parameters and master secret key. Then, the parameters and master secret key are given to adversary $\mathcal{A}$.

- Queries: Type-II adversary can query the following oracles.
  - Partial-Private-Key-Extraction: The challenger runs the Extract-Partial-Private-Key algorithm to generate partial private key. Then the partial private key is given to $\mathcal{A}$.

- Time-Key-Query: The challenger runs the Update-time-Key algorithm to generate the new time key. Then the challenger returns it to the adversary $\mathcal{A}$.
- Secret-Value-Query: The challenger runs Set-Secret-Value algorithm to generate the secret value, then returns it to the adversary $\mathcal{A}$.
- Public-Key-Request: For the adversary's query for public key, the challenger runs Set-Public-Key algorithm to generate public key for the adversary $\mathcal{A}$.
- Signature: The challenger runs Sign algorithm to generate the signature for the tuple of (M, ID, t).

• Forge: The adversary outputs a signature on the tuple (M^*, ID^*, T^*). And the tuple (M^*, ID^*, T^*) is the one that is never used in the Signature query.

3.5 Security Game for Type-III adversary

The challenger runs setup algorithm to generate public parameters and master secret key. Then, the parameters are given to adversary $\mathcal{A}$. Master secret key is kept secret.

• Queries: Type-III adversary can query the following oracles.
- Partial-Private-Key-Extraction: The challenger runs the Extract-Partial-Private-Key algorithm to generate partial private key. Then the partial private key is given to $\mathcal{A}$.
- Time-Key-Query: The challenger runs the Update-time-Key algorithm to generate the new time key. Then the challenger returns it to the adversary $\mathcal{A}$.
- Secret-Value-Query: The challenger runs Set-Secret-Value algorithm to generate the secret value, then returns it to the adversary $\mathcal{A}$.
- Public-Key-Request: For the adversary's query for public key, the challenger runs Set-Public-Key algorithm to generate public key for the adversary $\mathcal{A}$.
- Signature: The challenger runs Sign algorithm to generate the signature for the tuple of (M, ID, t).

• Forge: The adversary outputs a signature on the tuple (M^*, ID^*, T^*). And the tuple (M^*, ID^*, T^*) is the one that is never used in the Signature query.

3.6 Discrete Logarithm Problem

Given $a, g^a \in G$ where g is a generator, $a \in Z_p^*$, p is a prime and G is a certain group. The problem is to find the value of a. The discrete logarithm problem is assumed to be a computationally hard problem for the multiplicative group. For any probabilistic polynomial time algorithm, the advantage of the algorithm is negligibly small. The security of our proposed scheme relies on the difficulty of the discrete logarithm problem.

4. Proposed Scheme

This section describes the construction of the proposed scheme. Our proposed scheme contains eight probabilistic polynomial time algorithms. The security notations and descriptions used in the proposed scheme are described in Table 1.

• Setup: The KGC runs this algorithm to generate public parameters *params* and master secret key m_k. The KGC chooses $m_k \in Z_p^*$ and computes $pub_k = g^{m_k}$. Hash functions are $H_1 : \{0,1\}^* \to Z_p^*$, $H_2 : \{0,1\}^* \to Z_p^*$, $H_3 : \{0,1\}^* \to Z_p^*$, and $H_4 : \{0,1\}^* \to Z_p^*$. Then, the KGC publishes the parameters $params = \{pub_k, H_1, H_2, H_3, H_4\}$ and keeps master secret key secret.

Table 1 Notation and Description

Notation	Description
p	Large prime
g	Generator of Z_p^*
pub_k	Public key of KGC
m_k	Master Secret Key of KGC
PK_{ID}	Public key of user with ID
s_{ID}	Secret value of user with ID
(X_{ID}, D_{ID})	Partial private key of user with ID
t	Time
$D_{ID,t}$	Time key of user with ID
$SK_{ID,t}$	Private key of user with ID at time t
H_1, H_2, H_3, H_4	Collision-resistant one way hash functions
M	Message
σ	Signature

• Extract-Partial-Private-Key: This algorithm takes *params*, m_k and an identity ID as input and chooses $x_{ID} \in Z_p^*$ randomly. Then it computes $X_{ID} = g^{x_{ID}}$ and a partial private key $D_{ID} = H_1(ID)m_k + x_{ID} \mod p$. And then returns the (X_{ID}, D_{ID}) as the partial private key via secure channel.

• Update-time-Key: For each time t, the algorithm chooses $w_t \in Z_p^*$ randomly and computes $W_t = g^{w_t}$. Note that this algorithm computes W_t only once for all non-revoked users at each time t and KGC can pre-compute it. To generate new time key, it takes *params*, x_{ID}, D_{ID}, m_k and an identity ID as input and computes $D_{ID,t} = w_t + H_2(ID,t)x_{ID} + m_k H_3(ID,t,W_t) + D_{ID} \mod p$. Then $D_{ID,t}$ is transmitted to the user with ID via public channel.

• Set-Secret-Value: This algorithm takes *params* and an identity ID as input and produces a secret value $s_{ID} \in Z_p^*$ for the user with ID.

• Set-Private-Key: Taking *params*, s_{ID}, D_{ID}, and $D_{ID,t}$ as input, the user runs this algorithm. And the algorithm outputs $SK_{ID,t} = (D_{ID,t} - D_{ID}, s_{ID})$ as the private key.

• Set-Public-Key: The user with ID runs this algorithm $\mathcal{B}$ by taking *params* and s_{ID}. The output is public key $PK_{ID} = g^{s_{ID}}$.

• Sign: The sender with ID runs this algorithm to sign the message M. The algorithm takes *params*, ID, time t, $SK_{ID,t}$ and M as inputs and outputs the signature σ. The algorithm chooses $q \in Z_p^*$ randomly and calculates $Y = g^q$. The signature is calculated as follows:

$$\begin{aligned} m &= H_4(Y, t, M) \\ \theta &= m(D_{ID,t} - D_{ID} + s_{ID}) + q \mod p \end{aligned}$$

Then the signature $\sigma = (\theta, m)$ is set as output.

• Verify: The receivers run this algorithm to verify the signature by taking *params*, PK_{ID}, X_{ID}, t, M, W_t, and signature σ as input. Verification is done as follows:

$$\begin{aligned} h_2 &= H_2(ID, t) \\ h_3 &= H_3(ID, t, W_t) \\ Y &= g^{\theta}(pub_k^{h_3} W_t X_{ID}^{h_2} PK_{ID})^{-m} \\ m_v &= H_4(Y, t, M) \end{aligned}$$

If the m_v is equal to m, then the algorithm outputs "accept". Otherwise, it outputs "reject".

The correctness of the proposed scheme can be checked as follows:

$$\begin{aligned}
g^{\theta} &= g^{m(D_{ID,t}-D_{ID}+s_{ID})+q}\\
&= g^{m(w_t+H_2(ID,t)x_{ID}+m_kH_3(ID,t,W_t)+s_{ID})}g^q\\
&= g^{m(w_t+h_2x_{ID}+m_kh_3+s_{ID})}g^q\\
(pub_k^{h_3}W_tX_{ID}^{h_2}PK_{ID})^{-m} &= (g^{m_kh_3}g^{w_t}g^{x_{ID}h_2}g^{s_{ID}})^{-m}\\
&= g^{-m(m_kh_3+w_t+x_{ID}h_2+s_{ID})}\\
Y &= g^{\theta}(pub_k^{h_3}W_tX_{ID}^{h_2}PK_{ID})^{-m}\\
&= g^{m(w_t+h_2x_{ID}+m_kh_3+s_{ID})}g^qg^{-m(m_kh_3+w_t+x_{ID}h_2+s_{ID})}\\
&= g^q\\
&= Y
\end{aligned}$$

5. Security Proofs and Efficiency

In this section, we will show that the proposed signature scheme is secure in the random oracle model under the discrete logarithm problem. The four hash functions $H_i(i = 1, 2, 3, 4)$ are modelled as random oracles.

5.1 Security Proofs

Theorem 1. If Type I adversary can forge a RCLS scheme in probabilistic polynomial time with non-negligible probability, then there exists an algorithm $\mathcal{B}$ that can solve the discrete logarithm problem with non-negligible probability.

Proof: Let $\mathcal{B}$ has the tuple of (g, g^a) as the parameters for DLP problem. $\mathcal{B}$ plays as a challenger and $\mathcal{A}$ represents Type-I adversary. Adversary $\mathcal{A}$ is allowed to access $H_i(i = 1, 2, 3, 4)$ query for n times. The four hash functions are random oracles. $\mathcal{B}$ chooses $j \in [1, n]$ uniformly at random. Suppose the jth query is on (M^*, ID^*, T^*). All query and results are maintained in corresponding lists.

- Setup: Challenger $\mathcal{B}$ runs this algorithm to generate public parameters *params* and master secret key m_k. Then, the KGC publishes the parameters $params = \{pub_k, H_1, H_2, H_3, H_4\}$ and keeps master secret key secret.
- Queries: We assume that the adversary $\mathcal{A}$ always makes hash queries before requesting other queries. Adversary $\mathcal{A}$ can query the following queries using ID_i and time t.
  - $H_i(i = 1, 2, 3, 4)$ queries: For the hash query, $\mathcal{B}$ picks a random element from Z_p^* and returns it to $\mathcal{A}$.
  - Partial-Private-Key-Extraction: The query accepts identity ID_i as input. If $(i = j)$, the challenger $\mathcal{B}$ returns $X_{ID_i} = g^a$. Otherwise, $\mathcal{B}$ chooses $x'_{ID} \in Z_p^*$ randomly. Then it computes $X_{ID_i} = g^{x'_{ID}}$ and a partial private key $D_{ID} = H_1(ID_i)m_k + x'_{ID}\, mod\, p$. And then it returns (X_{ID_i}, D_{ID}) as the partial private key for $\mathcal{A}$.
  - Time-Key-Query: For the $\mathcal{A}$'s query for this query, if $(i = j)$, the game is aborted. Otherwise, the challenger $\mathcal{B}$ randomly chooses $w' \in Z_p^*$. Then, it produces $W_t = g^{w'}$. challenger $\mathcal{B}$ then generates time key $D_{ID_i,t} = w' + H_2(ID_i, t)x'_{ID} + m_kH_3(ID_i, t, W_t) + D_{ID}\, mod\, p$.
  - Secret-Value-Query: The challenger $\mathcal{B}$ randomly chooses $s'_{ID} \in Z_p^*$. Then, it returns s'_{ID} to $\mathcal{A}$.
  - Public-Key-Request: For the $\mathcal{A}$'s query for public key, the challenger $\mathcal{B}$ checks the secret key list. If the secret value is already in the list, $\mathcal{B}$ computes corresponding public key for the response. Otherwise, $\mathcal{B}$ randomly runs Set-Public-Key algorithm and returns public key to $\mathcal{A}$. And $\mathcal{B}$ adds the secret value to the secret value list.
  - Public-Key-Replace: This query allows the adversary to replace the public key with any new value.
  - Signature: For the signature query on (M, ID_i, t), if $(i = j)$, $\mathcal{B}$ picks $\theta', m' \in Z_p^*$ randomly and computes

$$\begin{aligned}
h_2 &= H_2(ID_i, t)\\
h_3 &= H_3(ID_i, t, W_t)\\
Y' &= g^{\theta'}(pub_k^{h_3}W_tX_{ID_i}^{h_2}PK_{ID_i})^{-m'}
\end{aligned}$$

    And $\mathcal{B}$ returns $\sigma' = (\theta', m')$. In case when $(i \neq j)$, it normally signs and outputs the signature.
- Forge: The adversary $\mathcal{A}$ outputs a signature σ^* on the tuple (M^*, ID^*, T^*).
- Analysis: Since the $H_i(i = 1, 2, 3, 4)$ are viewed as random oracles, $\mathcal{A}$ can get another signature $\sigma^{*'}$.

Suppose the adversary $\mathcal{A}$ can forge the signature scheme with advantage ϵ when running in time t' making q_{ip} partial private key queries, q_{pub} public key queries, q_{tk} time key update queries, q_{sign} signature queries, and $q_i(i = 1, 2, 3, 4)$ random oracle queries to $H_i(i = 1, 2, 3, 4)$ respectively. Then, there exists an algorithm $\mathcal{B}$ that can solve the DL problem with advantage $\epsilon' \geq \frac{\epsilon}{q_1}$ and running time $t'' = t' + (q_1 + q_2 + q_3 + q_4 + T_sq_{ip} + T_sq_{pub} + T_sq_{tk} + q_{sign})$ where T_s denotes the time for exponentiation. The running time of $\mathcal{B}$ is bounded by t''. Since discrete logarithm problem is computationally intractable in polynomial time, the scheme is secure against Type-I adversary.

Theorem 2. If Type II adversary can forge a RCLS scheme in probabilistic polynomial time with non-negligible probability, then there exists an algorithm $\mathcal{B}$ that can solve the discrete logarithm problem with non-negligible probability.

Proof: Let $\mathcal{B}$ has the tuple of (g, g^a) as the parameters for DLP problem. $\mathcal{B}$ plays as a challenger and $\mathcal{A}$ represents Type-II adversary. Adversary $\mathcal{A}$ is allowed to access $H_i(i = 1, 2, 3, 4)$ query for n times. The four hash functions are random oracles. $\mathcal{B}$ chooses $j \in [1, n]$ uniformly at random. Suppose the jth query is on (M^*, ID^*, T^*). All query and results are maintained in corresponding lists.

- Setup: Challenger $\mathcal{B}$ runs this algorithm to generate public parameters *params* and master secret key m_k. Then, the KGC publishes the parameters $params = \{pub_k, H_1, H_2, H_3, H_4\}$ and gives master secret key to $\mathcal{A}$. As the adversary $\mathcal{A}$ knows the master secret key, it can generate any partial private key.
- Queries: We assume that the adversary $\mathcal{A}$ always makes hash queries before requesting other queries. Adversary $\mathcal{A}$ can query the following queries using ID_i and time t.
  - $H_i(i = 1, 2, 3, 4)$ queries: For the hash query, $\mathcal{B}$ picks a random element from Z_p^* and returns it to $\mathcal{A}$.
  - Partial-Private-Key-Extraction: Challenger $\mathcal{B}$ generates partial private key for any ID.
  - Time-Key-Query: Challenger $\mathcal{B}$ generates the time key for any ID and outputs it.
  - Secret-Value-Query: The challenger $\mathcal{B}$ randomly chooses $s'_{ID} \in Z_p^*$. Then, it returns s'_{ID} to $\mathcal{A}$.
  - Public-Key-Request: For the $\mathcal{A}$'s query for public key, the challenger $\mathcal{B}$ checks whether $(i = j)$ or not. If it is equal, $\mathcal{B}$ returns $PK_{ID_i} = g^a$. Otherwise, $\mathcal{B}$ randomly runs Set-Public-Key algorithm and returns public key to $\mathcal{A}$. And $\mathcal{B}$ adds the secret value to the secret value list.

– Signature: For the signature query on (M, ID_i, t), if $(i = j)$, $\mathcal{B}$ picks $\theta^{'}, m^{'} \in Z_p^*$ randomly and computes

$$h_2 = H_2(ID_i, t)$$
$$h_3 = H_3(ID_i, t, W_t)$$
$$Y^{'} = g^{\theta^{'}}(pub_k^{h_3} W_t X_{ID_i}^{h_2} PK_{ID_i})^{-m^{'}}$$

And $\mathcal{B}$ returns $\sigma^{'} = (\theta^{'}, m^{'})$. In case when $(i \neq j)$, it normally signs and outputs the signature.

- Forge: The adversary $\mathcal{A}$ outputs a signature σ^* on the tuple (M^*, ID^*, T^*).
- Analysis: Since the $H_i(i = 1, 2, 3, 4)$ are viewed as random oracles, $\mathcal{A}$ can get another signature $\sigma^{*^{'}}$.

Suppose the adversary $\mathcal{A}$ can forge the signature scheme with advantage ϵ when running in time $t^{'}$, making q_{ip} partial private key queries, q_{pub} public key queries, q_{tk} time key update queries, q_{sign} signature queries, and $q_i(i = 1, 2, 3, 4)$ random oracle queries to $H_i(i = 1, 2, 3, 4)$ respectively. Then, there exists an algorithm $\mathcal{B}$ that can solve the DL problem with advantage $\epsilon^{'} \geq \frac{\epsilon}{q_2 q_3}$ and running time $t^{''} = t^{'} + (q_1 + q_2 + q_3 + q_4 + T_s q_{ip} + T_s q_{pub} + T_s q_{tk} + q_{sign})$ where T_s denotes the time for exponentiation. The running time of $\mathcal{B}$ is bounded by $t^{''}$. Since discrete logarithm problem is computationally intractable in polynomial time, the scheme is secure against Type-II adversary.

Theorem 3. If Type III adversary can forge a RCLS scheme in probabilistic polynomial time with non-negligible probability, then there exists an algorithm $\mathcal{B}$ that can solve the discrete logarithm problem with non-negligible probability.

Proof: Let $\mathcal{B}$ has the tuple of (g, g^a) as the parameters for DLP problem. $\mathcal{B}$ plays as a challenger and $\mathcal{A}$ represents Type-III adversary. Adversary $\mathcal{A}$ is allowed to access $H_i(i = 1, 2, 3, 4)$ query for n times. The four hash functions are random oracles. $\mathcal{B}$ chooses $j \in [1, n]$ uniformly at random. Suppose the jth query is on (M^*, ID^*, T^*). All query and results are maintained in corresponding lists.

- Setup: Challenger $\mathcal{B}$ runs this algorithm to generate public parameters *params* and master secret key m_k. Then, the KGC publishes the parameters $params = \{pub_k, H_1, H_2, H_3, H_4\}$ and keeps master secret key secret.
- Queries: We assume that the adversary $\mathcal{A}$ always makes hash queries before requesting other queries. Adversary $\mathcal{A}$ can query the following queries using ID_i and time t.

– $H_i(i = 1, 2, 3, 4)$ queries: For the hash query, $\mathcal{B}$ picks a random element from Z_p^* and returns it to $\mathcal{A}$.

– Partial-Private-Key-Extraction: Challenger $\mathcal{B}$ generates partial private key for any ID.

– Time-Key-Query: Challenger $\mathcal{B}$ generates the time key for any identity ID_i except $i = j$.

– Secret-Value-Query: The challenger $\mathcal{B}$ randomly chooses $s_{ID}^{'} \in Z_p^*$. Then, it returns $s_{ID}^{'}$ to $\mathcal{A}$.

– Public-Key-Request: For the $\mathcal{A}$'s query for public key, the challenger $\mathcal{B}$ checks the secret key list. If the secret value is already in the list, $\mathcal{B}$ computes corresponding public key for the response. Otherwise, $\mathcal{B}$ randomly runs Set-Public-Key algorithm and returns public key to $\mathcal{A}$. And $\mathcal{B}$ adds the secret value to the secret value list.

– Signature: For the signature query on (M, ID_i, t), if $(i = j)$, $\mathcal{B}$ picks $W_t = g^a, \theta^{'}, m^{'} \in Z_p^*$ randomly and computes

$$h_2 = H_2(ID_i, t)$$
$$h_3 = H_3(ID_i, t, W_t)$$
$$Y^{'} = g^{\theta^{'}}(pub_k^{h_3} W_t X_{ID_i}^{h_2} PK_{ID_i})^{-m^{'}}$$

And $\mathcal{B}$ returns $\sigma^{'} = (\theta^{'}, m^{'})$. In case when $(i \neq j)$, it normally signs and outputs the signature.

- Forge: The adversary $\mathcal{A}$ outputs a signature σ^* on the tuple (M^*, ID^*, T^*).
- Analysis: Since the $H_i(i = 1, 2, 3, 4)$ are viewed as random oracles, $\mathcal{A}$ can get another signature $\sigma^{*^{'}}$.

Suppose the adversary $\mathcal{A}$ can forge the signature scheme with advantage ϵ when running in time $t^{'}$, making q_{ip} partial private key queries, q_{pub} public key queries, q_{tk} time key update queries, q_{sign} signature queries, and $q_i(i = 1, 2, 3, 4)$ random oracle queries to $H_i(i = 1, 2, 3, 4)$ respectively. Then, there exists an algorithm $\mathcal{B}$ that can solve the DL problem with advantage $\epsilon^{'} \geq \frac{\epsilon}{q_2 q_3}$ and running time $t^{''} = t^{'} + (q_1 + q_2 + q_3 + q_4 + T_s q_{ip} + T_s q_{pub} + T_s q_{tk} + q_{sign})$ where T_s denotes the time for exponentiation. The running time of $\mathcal{B}$ is bounded by $t^{''}$. Since discrete logarithm problem is computationally intractable in polynomial time, the scheme is secure against Type-III adversary.

Table 2 Computational Cost Comparison

	Scheme [25]	Our Scheme
Sign	$1T_{expo}$	$1T_{expo}$
Verify	$4T_{expo}$	$4T_{expo}$
Time Key Update	vT_{expo}	T_{expo}

5.2 Computational Cost Comparison

Computational cost comparison is summarized in Table 2. Here, we denote the exponentiation operation as T_{expo}. In existing scheme [25], no pairing operation is required for signing. For verifying the signature, four exponentiation operations are required. Our proposed scheme requires one exponentiation operation and four exponentiation operations for both signing and verifying the signature respectively. Therefore, our proposed scheme and existing scheme [25] require the same number of computational operations for signing and verification. Although the schemes have same computation cost for signing and verification, the difference lies in the time key update of KGC. The number of non-revoked users is denoted as v. Although the existing scheme [25] requires vT_{expo} for key update, the computational cost of our proposed scheme is only $1T_{expo}$.

In our proposed scheme, time key can be delivered by using public channel like the existing schemes.

6. Conclusion

In this paper, we propose an efficient revocable certificateless signature scheme that reduces the number of exponentiation on key generation center (KGC). In signing and verifying the signature scheme, the proposed scheme avoids pairing operations. Moreover, time key distribution uses public channels. Compared with existing pairing free certificateless signature scheme, our proposed scheme requires only one exponentiation for time key generation at revoked time. We also show that the security of the proposed scheme under discrete logarithm assumption in the random oracles. For the future work, we intend to evaluate the actual computation load in real application environment.

Acknowledgments This research was supported in part by Grand-in-Aids for Scientific Research (B) numbered JP15H02702.

References

[1] ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms, in *IEEE transactions on information theory*, Vol.31, pp.469–472, IEEE (1985).

[2] Shamir, A.: Identity-based cryptosystems and signature schemes, in *Crypto*, Vol.84, pp.47–53 (1984).

[3] Al-riyami, S.S. and Paterson, K.G.: Certificateless public key cryptography, in *Asiacrypt*, Vol. 2894, pp.452–473 (2003).

[4] Micali, S.: Efficient certificate revocation, in *Technical Report MIT/LCS/TM-542b*, (1996).

[5] Micali, S.: Scalable Certificate Validation and Simplified PKI Management, in *PKI Research Workshop*, Vol.15 (2002).

[6] Boneh, D. and Franklin, M.: Identity-based encryption from the Weil pairing, in *Advances in Cryptography-CRYPTO*, pp.213–229, Springer Berlin (2001).

[7] Boldyreva, A., Goyal, V., and Kumar. V.: Identity-based encryption with efficient revocation, in *Proc. 15th ACM Conf. on computer and communications security*, pp.417–426, ACM (2008).

[8] Tseng, Y.M. and Tsai, T.T.: Efficient revocable ID-based encryption with a public channel, in *Computer J.* , No. 4, pp.475–486 (2011).

[9] Cheng, Z. and Comley, R.: Efficient Certificateless Public Key Encryption, in *IACR Cryptology ePrint Archive*, pp.12 (2005).

[10] Shi, Y. and Li, J.: Provable Efficient Certificateless Public Key Encryption, in *IACR Cryptography ePrint Archive*, pp.287 (2005).

[11] Baek, J., Safavi-Naini, R., and Susilo, W.: Certificateless public key encryption without pairing, in *Intl. Conf. on Information Security*, pp.134–148, Springer (2005).

[12] Sun, Y., Zhang, F., and Baek, J.: Strongly secure certificateless public key encryption without pairing, in *CANS*, Vol.4856, pp.194–208 (2007).

[13] Dent, A.W., Libert, B., and Paterson, K.G.: Certificateless encryption schemes strongly secure in the standard model, in *Intl. Workshop on Public Key Cryptography*, pp.344–359, Springer (2008).

[14] Gorantla, M.C. and Saxena, A.: An efficient certificateless signature scheme, in *Intl. Conf. on Computational and Information Science*, pp.110–116, Springer (2005).

[15] Zhang, Z., Wong, D.S., Xu, J., and Feng, D.: Certificateless public-key signature: security model and efficient construction, in *ACNS*, Vol.6, pp.293–308 (2006).

[16] Yap, W.S., Heng, S.H., and Goi, B.M.: An efficient certificateless signature scheme, in *Emerging Directions in Embedded and Ubiquitous Computing*, pp.322–331 (2006).

[17] Liu, W., Xie, Q., Wang, S., Han, L., and Hu, B.: Pairing-Free Certificateless Signature with Security Proof, in *J. of Computer Networks and Communications*, (2014).

[18] Ju, H.S., Kim, D.Y., Lee, D.H., Lim, J., and Chun, K.: Efficient revocation of security capability in certificateless public key cryptography, in *KES*, pp.453–459, LNCS 3683 (2005).

[19] Chow, S.S.M., Boyd, C., Nieto, J.M.G.: Security-Mediated Certificateless Cryptography, in *PKC*, pp.508–524, LNCS 3958 (2006)

[20] Yap, W.S., Chow, S.S.M., Heng, S.H., and Goil, B.M.: Security Mediated Certificateless Signatures, in *ACNS*, pp.459–477, LNCS 4521 (2007).

[21] Tsai, T.T., Huang, S.S., and Tseng, Y.M.: Secure certificateless signature with revocation in the standard model, in *Mathematical Problems in Engineering*, (2014).

[22] Sun, Y., Zhang, F., Shen, L., and Deng, R.H.: A Revocable Certificateless Signature Scheme, in *J. of Computers*, Vol.9, No.8, pp.1843–1850 (2014).

[23] Tseng, Y.M., Tsai, T.T., Huang, S.S., and Hung, Y.H.: Strongly Unforgeable Revocable Certificateless Signature, in *Proc. 4th Intl. Conf. on Informatics and Applications*, pp.18–31 (2015).

[24] Shun, Y. and Shen, L.: Pairing-Free and Revocable Certificateless Signature Against Signing Key Exposure, in *J. of Emerging Trends in Computing and Information Sciences*, Vol.5, No.11 (2014).

[25] Shun, Y., Zhuoran, Z., and Shen, L.: A Revocable Certificateless Signature Scheme Without Pairing, in *Intl. Conf. on Cloud Computing and Security*, pp.355–36, Springer (2016).