INFORMATICA, 2016, Vol. 27, No. 3, 549–572
© 2016 *Vilnius University*
DOI: http://dx.doi.org/10.15388/Informatica.2016.99

A Revocable Certificateless Short Signature Scheme and Its Authentication Application

Ying-Hao HUNG, Yuh-Min TSENG*, Sen-Shan HUANG
Department of Mathematics, National Changhua University of Education Jin-De Campus Chang-Hua City 500, Taiwan
e-mail: ymtseng@cc.ncue.edu.tw

Received: July 2015; accepted: February 2016

Abstract. Certificateless short signature (CLSS) possesses the advantages of both certificateless signature and short signature. CLSS eliminates the certificate management in conventional signatures and solves the key escrow problem in ID-based signatures. In the meantime, due to its short signature length, CLSS reduces the bandwidth for communication so that it is suitable for some specific authentication applications requiring bandwidth-constrained communication environments. However, up to now, there is no work on studying the revocation problem in existing CLSS schemes. In this article, we address the revocation problem and propose the first revocable certificateless short signature (RCLSS) scheme. Based on the computational Diffie–Hellman (CDH) assumption, we demonstrate that our RCLSS scheme possesses strong unforgeability against adaptive chosen-message attacks under an accredited security model. It turns out that our scheme has the shortest signature length while retaining computational efficiency. Thus, the proposed RCLSS scheme is well suited for low-bandwidth communication environments. Finally, we combine the proposed RCLSS scheme with cloud revocation authority (CRA) to present a CRA-aided authentication scheme with period-limited privileges for mobile multi-server environment.

Key words: certificateless signature, short signature, revocation, unforgeability, authentication.

1. Introduction

Shamir (1984) introduced the notion of identity-based cryptography (IBC). In IBC, a user's public key is derived from her/his identity information such as e-mail address and physical IP address and so on. And the corresponding private key is generated by a trusted private key generator (PKG). The user's private key is given to the user via a secure channel and its legitimacy can be verified publicly by her/his identity information. As opposed to conventional public key systems, IBC eliminates the requirement of certificates. However, IBC inevitably suffers from the key escrow problem, namely, the PKG knows all the users' private keys. In such a case, the PKG can decrypt any ciphertext or forge a signature on any message on behalf of any user. To solve the key escrow problem in IBC and to eliminate the use of certificates in conventional public key cryptography, Al-Riyami and Paterson (2003) proposed the notion of certificateless public key cryptography (CL-PKC).

*Corresponding author.

In CL-PKC, a semi-trusted third party, called the key generation centre (KGC), is responsible to generate the initial key for a user. The full private key of a user is the combination of her/his initial key and a secret value chosen by the user. Even though the KGC knows the initial key, it has no access to the full private key of the user. Hence, the key escrow problem is resolved. Meanwhile, the user's public key is independently generated and published by the user. Since no explicit certification of a public key is required, CL-PKC eliminates the use of certificates. Afterwards, the study of CL-PKC has received significant attention from researchers and numerous cryptographic primitives have been presented such as certificateless public-key encryption (CL-PKE) (Libert and Quisquater, 2006; Dent, 2008; Yang and Tan, 2011) and certificateless signature (CLS) (Huang *et al.*, 2005; Hu *et al.*, 2006; Zhang and Zhang, 2008).

1.1. *Related Work*

Al-Riyami and Paterson (2003) presented a security model for certificateless public key cryptography. There are two types of adversaries in their model, namely, Type I adversary (outsiders) and Type II adversary (honest-but-curious KGC). Although Al-Riyami and Paterson presented a concrete CLS scheme, they did not offer the security notions for CLS schemes. Moreover, in CL-PKC, there is no certificate to authenticate a user's public key. In this situation, an adversary is able to replace a user's public key with a fake key of its choice. This is known as the public key replacement attack. Huang *et al.* (2005) first defined formal security notions for CLS schemes and pointed out that Al-Riyami and Paterson's CLS scheme suffers from public key replacement attacks by outsiders. The public key replacement attacks means that an outsider can replace the public key of a signer and forge valid signatures on any message on behalf of the signer, without knowing the corresponding initial key. Hu *et al.* (2006) presented a fairly strong security model for CLS schemes and their security model is generally adopted to formalize the security of CLS schemes. To improve the efficiency and security of CLS schemes, Zhang and Zhang (2008) proposed a secure CLS scheme against key replacement attacks. Furthermore, several CLS constructions (Xiong *et al.*, 2008; Yu *et al.*, 2012; Cheng *et al.*, 2013; Hung *et al.*, 2015) in the standard model have been proposed to remove the usage of random oracles (Bellare and Rogaway, 1993). Short signature aims at the reduction of communication bandwidth, and it is suitable for specific applications requiring bandwidth-constrained communication environments such as mobile communication. To provide relatively short signature, the first certificateless short signature (CLSS) scheme was proposed by Huang *et al.* (2007). Huang *et al.* also modified the security model for certificateless signature schemes, and categorized Types I/II adversaries into three kinds, namely, normal, strong and super Types I/II adversaries according to their attacking capabilities. A super adversary is more powerful than the others since it is bound by the least querying restrictions and can obtain the valid signatures without knowing the secret value of the replaced public key. However, Shim (2009) demonstrated that Huang *et al.*'s CLSS scheme is insecure. Afterwards, Du and Wen (2009) proposed an improved CLSS scheme and claimed that their scheme is provably secure against the strong Type I and normal Type II adversaries. Unfortunately, Choi *et al.* (2011) demonstrated that Du and Wen's scheme is insecure against the strong Type I adversary. They also proposed an improvement and claimed that their scheme is secure against the super Types I/II adversary. Tso *et al.* (2012) also proposed a new and efficient CLSS scheme. However, Du and Wen (2014) demonstrated that both schemes of Choi *et al.* and Tso *et al.* are insecure against the strong Type I adversary. Recently, to enhance the security of CLSS, Chen *et al.* (2013) presented a survey article for CLS and CLSS schemes while proposing a secure CLSS scheme.

1.2. *Contributions*

A public key setting must provide a revocation mechanism to revoke illegal or compromised users from the system. Several situations require the public key of a user to be revoked before its intended expiration date. In the conventional public key settings, a well-known revocation approach is to maintain the so-called certificate revocation list (CRL) (Housley *et al.*, 2002). In the CRL approach, when receiving a public key and its associated certificate, the user first validates them and then looks up the CRL to ensure that the public key has not been revoked. Due to the lack of the usage of certificates, the CRL approach is no longer suited for the certificateless public key setting. Recently, Tsai and Tseng (2015) and Shen *et al.* (2013), independently, proposed a revocable certificateless public-key encryption (RCL-PKE) scheme based on Tseng and Tsai's revocable ID-based encryption with public channels (Tseng and Tsai, 2012). Tsai *et al.* (2014a, 2014b) further proposed a revocable hierarchical ID-based encryption scheme. Afterwards, Sun *et al.* (2014) and Tsai *et al.* (2014a, 2014b), respectively, proposed a revocable certificateless signature (RCLS) scheme in the random oracle model and in the standard model. However, the signature sizes of all the RCLS schemes are more than one group element. Up to now, no revocable certificateless short signature (RCLSS) scheme is proposed and the design of RCLSS scheme remains an interesting and challenging problem. In this article, we address the revocation problem and propose the first RCLSS scheme. The proposed RCLSS scheme possesses three merits.

(1) Under the standard computational Diffie–Hellman (CDH) assumption, our RCLSS scheme is provably secure against the attacks of super Types I/II adversary under an accredited security model (Huang *et al.*, 2007).
(2) Our RCLSS scheme provides a public revocation mechanism to revoke misbehaving/compromised users.
(3) Our scheme enjoys lower communication bandwidth while retaining computation efficiency as compared with the previously proposed RCLS scheme.

To demonstrate an efficient application of our RCLSS scheme, we introduce the role of cloud revocation authority (CRA) and present a CRA-aided authentication scheme with period-limited privileges for multi-server environments.

1.3. *Organization*

The rest of the paper is organized as follows. Preliminaries are given in Section 2. We formally present the framework and security notions for RCLSS schemes in Section 3.

A concrete RCLSS scheme is proposed in Section 4. We analyse the security of the proposed RCLSS scheme in Section 5. In Section 6, we make performance analysis and comparisons. In Section 7, a CRA-aided authentication scheme with period-limited privileges for multi-server environment is proposed. Conclusions are given in Section 8.

2. Preliminaries

In this section, we review some fundamental backgrounds required in this article, namely, bilinear pairings and security assumption.

2.1. *Bilinear Pairings*

Let $\mathbb{G}$ denote an additive group of prime order q and $\mathbb{G}_T$ be a multiplicative group of the same order. Let $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be an admissible bilinear mapping with the following properties.

(1) Bilinearity: for all $P, Q \in \mathbb{G}$ and $x, y \in Z_q^*$, we have $\hat{e}(xP, yQ) = \hat{e}(P, Q)^{xy}$.
(2) Non-degeneracy: there exist $P, Q \in \mathbb{G}$ such that $\hat{e}(P, Q) \neq 1$.
(3) Efficient computability: for all $P, Q \in \mathbb{G}$, it is efficient to compute $\hat{e}(P, Q)$.

2.2. *Security Assumption*

Here, we present a hard mathematical problem and its associated security assumption.

- *Computational Diffie–Hellman* (CDH) *problem*: given $P, aP, bP \in \mathbb{G}$ with uniformly random choices of $a, b \in Z_q^*$, the CDH problem in $\langle \mathbb{G} \rangle$ is to compute abP.

Definition 1. *(Computational Diffie–Hellman (CDH) assumption).* For $P, aP, bP \in \mathbb{G}$, we say that the CDH assumption in $\langle \mathbb{G} \rangle$ holds if no probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ can compute abP with non-negligible advantage. Here, the advantage of $\mathcal{A}$ is defined as $Adv^{CDH} = \Pr[\mathcal{A}(P, aP, bP) = abP]$.

3. Framework and Adversarial Model of RCLSS

3.1. *Framework*

The framework of RCLSS scheme is identical to that of the RCLS schemes in Sun *et al.* (2014) and Tsai *et al.* (2014a, 2014b). A RCLSS scheme consists of eight algorithms, namely, *Setup*, *Initial key extract*, *Time key update*, *Set secret value*, *Set private key*, *Set public key*, *Sign and Verify* algorithms.

- *Setup*: this algorithm is a probabilistic algorithm run by the key generation centre (KGC) that takes a security parameter as input, and returns the master secret key s and the public parameters *PP*. *PP* is made public and available for all the other algorithms.

- *Initial key extract*: this algorithm is a probabilistic algorithm run by the KGC that takes a user's identity ID as input, and outputs the user's initial key d_{ID} and the first partial public key R_{ID}.
- *Time key update*: this algorithm is a deterministic algorithm run by the KGC that takes as input a time period t, the master secret key s and a user's identity ID, and returns the user's time update key $T_{ID,t}$.
- *Set secret value*: this algorithm is a probabilistic algorithm run by a user that takes the user's identity ID as input, randomly selects a secret value x_{ID}, computes the second partial public key P_{ID}, and then returns x_{ID} and P_{ID}.
- *Set private key*: this algorithm is a deterministic algorithm run by a user that takes as input the user's initial key d_{ID}, time update key $T_{ID,t}$ and secret value x_{ID}, and returns a private key $sk_{ID,t} = (d_{ID}, T_{ID,t}, x_{ID})$.
- *Set public key*: this algorithm is a deterministic algorithm run by a user that takes as input the user's two partial public keys R_{ID} and P_{ID}, and returns the full public key $PK_{ID} = (R_{ID}, P_{ID})$.
- *Sign*: given a time period t, this algorithm is a deterministic algorithm run by a user that takes as input the user's identity ID, private key $sk_{ID,t}$ and a message m, and returns a signature σ.
- *Verify*: this algorithm is a deterministic algorithm that takes as input a time period t, a message m, a signature σ, a user identity ID with PK_{ID}, and outputs either "accept" or "reject".

3.2. *Security Model*

According to the security models in the CLS schemes (Al-Riyami and Paterson, 2003; Hu *et al.*, 2006), there are two types of adversaries with different capabilities. Nevertheless, Huang *et al.* (2007) categorized the adversaries of each type into three levels, referred as normal, strong, and super adversaries. In Huang *et al.*'s scheme, a super adversary is bound by the least querying restrictions. A super adversary can obtain the valid signatures without knowing the secret value of the replaced public key. Based on the security notions for CLS schemes above (Hu *et al.*, 2006; Huang *et al.*, 2007), Sun *et al.* (2014) and Tsai *et al.* (2014a, 2014b) added a new type of adversary (a revoked user) in the security notions for revocable certificateless signature (RCLS) scheme. We present three kinds of adversaries, namely, Type I (outsider), Type II (honest-but-curious KGC) and Type III (revoked user). A Type I adversary A_I acts as an outsider who can replace the public key of any entity, except the target entity, with a value of her/his choice, but cannot access the master secret key. A Type II adversary A_{II} models an honest-but-curious KGC that has access to the master secret key, but cannot perform any public key replacement. A revoked user (A_{III} adversary) cannot obtain the current time update key $T_{ID,t}$, but still holds her/his initial key d_{ID} and has the ability to replace the public key of any entity, except the target entity, with a value of her/his choice. The security notions for RCLSS schemes are modelled by the following three games (Games I, II, III), which simulate the interactions between a challenger $\mathcal{C}$ and three types of super adversaries, respectively.

Definition 2. A secure RCLSS scheme possesses strong unforgeability against adaptive chosen-message attacks (UF-RCLSS-ACMA) if no probabilistic polynomial-time adversary has a non-negligible advantage in the following UF-RCLSS-ACMA games (Games I, II, III) played with a challenger.

Game 1 (for outsider, A_I)

– *Setup*. The challenger $\mathcal{C}$ runs the *Setup* algorithm to produce a master secret key s and a list of public parameters PP. PP is given to the adversary A_I and s is kept secret by the challenger $\mathcal{C}$.
– *Queries*. The adversary A_I may make a number of different queries to the challenger $\mathcal{C}$ in an adaptive manner as follows.
  - *Initial key extract* (ID). The challenger $\mathcal{C}$ runs the *Initial key extract* algorithm and returns the user's initial key d_{ID} and the first partial public key R_{ID} to A_I.
  - *Time key update* (ID, t). $\mathcal{C}$ runs the *Time key update* algorithm and returns the user's time update key $T_{ID,t}$ to A_I.
  - *Set secret value* (ID). Upon receiving this query, $\mathcal{C}$ randomly selects a secret value x_{ID}, and computes the second partial public key $P_{ID} = x_{ID} \cdot P$ and returns x_{ID} and P_{ID} to A_I.
  - *Public key retrieve* (ID). Upon receiving this query, $\mathcal{C}$ runs the *Set secret value* and *Initial key extract* algorithms to obtain P_{ID} and R_{ID}, and returns the user's full public key $PK_{ID} = (R_{ID}, P_{ID})$ to A_I.
  - *Public key replacement* (ID, PK'_{ID}). The adversary A_I chooses a new public key PK'_{ID} for the user with identity ID. The challenger $\mathcal{C}$ records this replacement. Note that, after replacement, the new public key PK'_{ID} will be used by $\mathcal{C}$ in all further computations and responses to $A_I{}'s$ requests.
  - *Super sign* (m, ID, t). Upon receiving this query on (m, ID, t) under the current public key $PK_{ID} = (R_{ID}, P_{ID})$, $\mathcal{C}$ runs the *Super sign* algorithm to generate a valid signature σ and returns it to A_I.
– Forgery. The adversary A_I generates a tuple $(m^*, ID^*, t^*, \sigma^*, PK_{ID^*})$. Here PK_{ID^*} is the original public key without being replaced. We say that the adversary A_I wins Game I if the following conditions are satisfied.
  (1) The response of the *Verify* algorithm on $(m^*, ID^*, t^*, \sigma^*, PK_{ID^*})$ is "accept".
  (2) (m^*, ID^*, t^*) has never been submitted during the *Super sign* query.
  (3) ID^* has never been submitted during the *Initial key extract* query.

In the sequel, the advantage of A_I is defined as the probability that A_I wins Game I.

Game 2 (for honest-but-curious KGC, A_{II})

– *Setup*. The challenger $\mathcal{C}$ runs the *Setup* algorithm to produce a master secret key s and a list of public parameters PP. PP and s are sent to A_{II}.
– *Queries*. Since the adversary A_{II} knows the master secret key, it can compute all the initial keys and time update keys. The adversary A_{II} may make a number of queries as in **Game I**, except the *Initial key extract* and *Time key update* queries.

- *Forgery*. The adversary A_{II} generates a tuple $(m^*, ID^*, t^*, \sigma^*, PK_{ID^*})$. Here PK_{ID^*} is the original public key without being replaced. We say that the adversary A_{II} wins Game II if the following conditions are satisfied.
  (1) The response of the *Verify* algorithm on $(m^*, ID^*, t^*, \sigma^*, PK_{ID^*})$ is "accept".
  (2) (m^*, ID^*, t^*) has never been submitted during the *Super sign* query.
  (3) ID^* has never been submitted during the *Set secret value* query.

Game 3 (for revoked user, A_{III})

- *Setup*. The challenger $\mathcal{C}$ runs the *Setup* algorithm to produce a master secret key s and a list of public parameters PP. PP is given to A_{III} and s is kept secret by $\mathcal{C}$.
- *Queries*. The adversary A_{III} may make a number of queries as in **Game I** in an adaptive manner.
- *Forgery*. The adversary A_{III} generates a tuple $(m^*, ID^*, t^*, \sigma^*, PK_{ID^*})$. Here PK_{ID^*} is the original public key without being replaced. We say that the adversary A_{III} wins Game III if the following conditions are satisfied.
  (1) The response of the *Verify* algorithm on $(m^*, ID^*, t^*, \sigma^*, PK_{ID^*})$ is "accept".
  (2) (m^*, ID^*, t^*) has never been submitted during the *Super sign* query.
  (3) (ID^*, t^*) has never been submitted during the *Time update key* query.

In the sequel, the advantage of A_{III} is defined as the probability that A_{III} wins Game III.

4. A Concrete RCLSS Scheme

As mentioned in Section 3.1, the proposed RCLSS scheme consists of eight algorithms, namely, *Setup, Initial key extract, Time key update, Set secret value, Set private key, Set public key, Sign* and *Verify* algorithms.

- *Setup*: given a security parameter l, the KGC first generates two groups $\mathbb{G}$ and $\mathbb{G}_T$ of prime order $q > 2^l$, an admissible bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, and an arbitrary generator P of $\mathbb{G}$. Next, the KGC randomly chooses a master secret key $s \in \mathbb{Z}_q^*$ and computes $P_{pub} = s \cdot P$ as the system public key. Finally, the KGC picks four hash functions $f : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$, and H_0, H_1, $H_2 : \{0, 1\}^* \rightarrow \mathbb{G}$. The public parameters are presented as $PP = \langle \mathbb{G}, \mathbb{G}_T, q, \hat{e}, P, P_{pub}, f, H_0, H_1, H_2 \rangle$.
- *Initial key extract*: given a user's identity $ID \in \{0, 1\}^*$, the KGC randomly chooses $r_{ID} \in \mathbb{Z}_q^*$, and computes the first partial public key $R_{ID} = r_{ID} \cdot P$ and its corresponding initial key $d_{ID} = r_{ID} + s \cdot f(ID, R_{ID})$. The KGC returns R_{ID} and d_{ID} to the user via a secure channel.
- *Time key update*: given a non-revoked user's identity ID and a period t, the KGC computes the user's time update key $T_{ID,t} = s \cdot H_0(ID, t)$, and sends $T_{ID,t}$ to the user via a public channel.
- *Set secret value*: a user with identity ID randomly selects a secret value $x_{ID} \in \mathbb{Z}_q^*$ and computes the second partial public key $P_{ID} = x_{ID} \cdot P$.

- *Set private key*: a user with identity *ID* sets her/his full private key $sk_{ID,t}$ as $(d_{ID}, T_{ID,t}, x_{ID})$ for the period t.
- *Set public key*: a user with R_{ID} and P_{ID} sets her/his (full) public key PK_{ID} as (R_{ID}, P_{ID}).
- *Signing*: in a period t, given a message m, a non-revoked signer with identity *ID* first computes $T_1 = H_1(m, ID, PK_{ID}, P_{pub}, t)$, and $T_2 = H_2(m, ID, PK_{ID}, P_{pub}, t)$. The signer then uses the private key $sk_{ID}, t = (d_{ID}, T_{ID,t}, x_{ID})$ to generate the signature $\sigma = x_{ID} \cdot T_1 + d_{ID} \cdot T_2 + T_{ID,t}$.
- *Verify*: a verifier validates a given signature tuple $(m, ID, PK_{ID} = (R_{ID}, P_{ID}), t, \sigma)$ by the following two steps.
  (1) Compute $h = f(ID, R_{ID})$, $T_0 = H_0(ID, t)$, $T_1 = H_1(m, ID, PK_{ID}, P_{pub}, t)$ and $T_2 = H_2(m, ID, PK_{ID}, P_{pub}, t)$.
  (2) Verify the equality $\hat{e}(P, \sigma) = \hat{e}(P_{ID}, T_1) \cdot \hat{e}(R_{ID} + h \cdot P_{pub}, T_2) \cdot \hat{e}(P_{pub}, T_0)$. If it holds, output "accept". Otherwise, output "reject".

5. Security Analysis

In the proposed RCLSS scheme, a user's full private key consists of three components, namely, the initial key, the time update key and the secret value. As the aforementioned UF-RCLSS-ACMA games in Definition 2, there are three kinds of adversaries, which include Type I (outsider), Type II (honest-but-curious KGC) and Type III (revoked user). In the following, we demonstrate that the proposed RCLSS is provably secure against Types I, II and III adversaries in Theorems 1, 2 and 3, respectively.

Theorem 1. *The proposed RCLSS scheme is provably secure against the UF-RCLSS-ACMA attack of Type I adversary (outsider). Concretely, assume that an UF-RCLSS-ACMA adversary $\mathcal{A}$ with a non-negligible advantage ϵ can break the proposed RCLSS scheme within running time τ. Then, we can construct an algorithm $\mathcal{C}$ that can solve the CDH problem with a non-negligible advantage*

$$\epsilon' \geqslant (1 - 1/q_e)\big(1 - 1/(1 + q_{ss})\big)^{q_{ss}}\big(1/q_e(1 + q_{ss})\big)\epsilon$$

within running time $\tau' = \tau + O(q_0 + q_1 + q_2 + q_e + q_{sv} + q_{ss})\tau_1$, where, in running time τ', $\mathcal{A}$ may ask at most q_i queries to the random oracles H_i $(i = 0, 1, 2)$, q_e queries to the Initial key extract *oracle, q_{sv} queries to the* Secret value extract *oracle and q_{ss} queries to the* Super sign *oracle, respectively; τ_1 is the time needed to perform a scalar multiplication in $\mathbb{G}$.*

Proof. Given a random instance (P, aP, bP) of the CDH problem in $\mathbb{G}$, where P, aP, $bP \in \mathbb{G}$ with unknown $a, b \in \mathbb{Z}_q^*$, we would like to construct an algorithm $\mathcal{C}$ to output the CDH solution abP. Here, the algorithm $\mathcal{C}$ will play the challenger in Game I and interacts with the adversary $\mathcal{A}$ as follows.

– *Setup*. The challenger $\mathcal{C}$ sets $P_{pub} = aP$ and the public parameters $PP = \langle \mathbb{G}, \mathbb{G}_T, q, \hat{e}, P, P_{pub}, f, H_0, H_1, H_2 \rangle$, where the hash functions f, H_0, H_1 and H_2 are random oracles controlled by $\mathcal{C}$. Finally, $\mathcal{C}$ returns PP to $\mathcal{A}$.
– *Queries*. $\mathcal{C}$ first chooses a random $i \in [1, q_e]$ and sets ID' as the target identity of the i-th query to the *Initial key extract* oracle. $\mathcal{C}$ answers the queries issued by $\mathcal{A}$ as follows.
  - *f queries*: at any time, $\mathcal{A}$ can issue queries with (ID, R_{ID}) to the random oracle f. To respond these queries, $\mathcal{C}$ maintains an initially-empty list L_f of tuples of the form $\langle ID, R_{ID}, h \rangle$ and responds as follows.
    1. If (ID, R_{ID}) already appears in L_f with the tuple $\langle ID, R_{ID}, h \rangle$, then $\mathcal{C}$ responds with h.
    2. Otherwise, the challenger $\mathcal{C}$ randomly chooses $h \in \mathbb{Z}_q^*$, adds $\langle ID, R_{ID}, h \rangle$ in L_f, and responds with h.
  - *H_0 queries*: at any time, $\mathcal{A}$ can issue queries with (ID, t) to the random oracle H_0. To respond these queries, $\mathcal{C}$ maintains an initially-empty list L_0 of tuples of the form $\langle ID, t, u, T_0, T_{ID,t} \rangle$ and responds as follows.
    1. If (ID, t) already appears in L_0 with a tuple $\langle ID, t, u, T_0, T_{ID,t} \rangle$, then $\mathcal{C}$ responds with T_0.
    2. Otherwise, $\mathcal{C}$ randomly chooses $u \in \mathbb{Z}_q^*$, sets $T_0 = u \cdot P$, $T_{ID,t} = u \cdot P_{pub}$, and adds the tuple $\langle ID, t, u, T_0, T_{ID,t} \rangle$ in L_0. Then, $\mathcal{C}$ responds with T_0.
  - *H_1 queries*: at any time, $\mathcal{A}$ can issue queries with (m, ID, PK_{ID}, t) to the random oracle H_1. To respond these queries, $\mathcal{C}$ maintains an initially-empty list L_1 of tuples of the form $\langle m, ID, PK_{ID}, t, w, T_1 \rangle$ and responds as follows.
    1. If (m, ID, PK_{ID}, t) already appears in L_1 with a tuple $\langle m, ID, PK_{ID}, t, w, T_1 \rangle$, then $\mathcal{C}$ responds with T_1.
    2. Otherwise, $\mathcal{C}$ randomly chooses $w \in \mathbb{Z}_q^*$, sets $T_1 = w \cdot P$, and adds $\langle m, ID, PK_{ID}, t, w, T_1 \rangle$ in L_1. Then, $\mathcal{C}$ responds with T_1.
  - *H_2 queries*: at any time, $\mathcal{A}$ can issue queries with (m, ID, PK_{ID}, t) to the random oracle H_2. To respond these queries, $\mathcal{C}$ maintains an initially-empty list L_2 of tuples of the form $\langle m, ID, PK_{ID}, t, v, T_2, coin \rangle$ and responds as follows.
    1. If (m, ID, PK_{ID}, t) already appears in L_2 with a tuple $\langle m, ID, PK_{ID}, t, v, T_2, coin \rangle$, then $\mathcal{C}$ responds with T_2.
    2. Otherwise, we split into two cases. If $ID \neq ID'$, $\mathcal{C}$ randomly chooses $v \in \mathbb{Z}_q^*$ and sets $T_2 = v \cdot P$ and $coin = 0$. If $ID = ID'$, $\mathcal{C}$ first flips a random $coin \in \{0, 1\}$ with $\Pr[coin = 1] = \delta$ for some δ that will be determined later. $\mathcal{C}$ sets $T_2 = v \cdot P$ if $coin = 0$, and $T_2 = v \cdot bP$ if $coin = 1$. Finally, $\mathcal{C}$ adds $\langle m, ID, PK_{ID}, t, v, T_2, coin \rangle$ in L_2 and responds with T_2.
  - *Initial key extract queries*: upon receiving this query with ID, $\mathcal{C}$ maintains a list K^{list} of tuples of the form $\langle ID, d_{ID}, R_{ID}, h \rangle$ and responds as follows.
    1. If ID already appears in K^{list} with a tuple $\langle ID, d_{ID}, R_{ID}, h \rangle$, then $\mathcal{C}$ responds with the initial key d_{ID} and the first partial public key R_{ID}.
    2. Otherwise, we split into two cases. If $ID = ID'$, $\mathcal{C}$ returns failure and terminates. If $ID \neq ID'$, $\mathcal{C}$ randomly chooses $h, z \in \mathbb{Z}_q^*$, and sets $R_{ID} = z \cdot P - h \cdot$

P_{pub}, $d_{ID} = z$ with $f(ID, R_{ID}) = h$. (Note that these choices of the first partial public key R_{ID}, the initial key d_{ID} and $f(ID, R_{ID})$ meets the definitions in our RCLSS scheme in Section 4.) Moreover, if this tuple $\langle ID, R_{ID}, h\rangle$ already exists in L_f, $\mathcal{C}$ needs to randomly pick $h, z \in \mathbb{Z}_q^*$ repeatedly until the tuple is new. Finally, $\mathcal{C}$ adds the tuple $\langle ID, d_{ID}, R_{ID}, h\rangle$ in K^{list} and responds with d_{ID} and R_{ID}.

- *Set secret value queries*: upon receiving this query with ID, $\mathcal{C}$ maintains a list P^{list} of tuples of the form $\langle ID, x_{ID}, P_{ID}\rangle$ and responds as follows.
  1. If ID already appears in the list P^{list} with a tuple $\langle ID, x_{ID}, P_{ID}\rangle$, then $\mathcal{C}$ responds with the secret value x_{ID} and the second partial public key P_{ID}.
  2. Otherwise, $\mathcal{C}$ randomly chooses $x_{ID} \in \mathbb{Z}_q^*$, and computes $P_{ID} = x_{ID} \cdot P$. Finally, $\mathcal{C}$ adds the tuple $\langle ID, x_{ID}, P_{ID}\rangle$ in P^{list} and responds with x_{ID} and P_{ID}.
- *Public key retrieve queries*: upon receiving this query with ID, $\mathcal{C}$ maintains a list PK^{list} of tuples of the form $\langle \mathrm{ID}, x_{ID}, R_{ID}, P_{ID}\rangle$ and responds as follows.
  1. If ID already appears in PK^{list} with a tuple $\langle ID, x_{ID}, P_{ID}, R_{ID}\rangle$, then $\mathcal{C}$ responds with the public key $PK_{ID} = (R_{ID}, P_{ID})$.
  2. Otherwise, $\mathcal{C}$ issues the *Initial key extract* and *Set secret value* queries to obtain R_{ID} and P_{ID}, and adds the tuple $\langle ID, x_{ID}, P_{ID}, R_{ID}\rangle$ in PK^{list}. Then, $\mathcal{C}$ responds with the public key $PK_{ID} = (R_{ID}, P_{ID})$. Note that if $ID = ID'$, the challenger $\mathcal{C}$ randomly chooses $h, r_{ID} \in \mathbb{Z}_q^*$, and sets $R_{ID} = r_{ID} \cdot P$, $h = f(ID, R_{ID})$ and $d_{ID} = \perp$. The challenger $\mathcal{C}$ stores the tuple $\langle ID, d_{ID}, P_{ID}, h\rangle$ in K^{list} in advance when setting ID as the target identity before performing all queries.
- *Public key replace queries*: upon receiving this query with $(ID, PK'_{ID} = (R'_{ID}, P'_{ID}))$, the challenger $\mathcal{C}$ replaces, respectively, the tuple $\langle ID, x_{ID}, R_{ID}, P_{ID}\rangle$ in PK^{list} with $\langle ID, \perp, R'_{ID}, P'_{ID}\rangle$ and the tuple $\langle ID, x_{ID}, P_{ID}\rangle$ in P^{list} with $\langle ID, \perp, P'_{ID}\rangle$.
- *Time key update queries*: upon receiving this query with (ID, t), $\mathcal{C}$ looks up the corresponding tuple $\langle ID, t, u, T_0, T_{ID,t}\rangle$ in L_0. If the tuple appears in L_0, $\mathcal{C}$ responds with $T_{ID,t}$. Otherwise, $\mathcal{C}$ issues the H_0 query to obtain $T_{ID,t} = s \cdot H_0(ID, t)$ and returns it to $\mathcal{A}$.
- *Signing queries*: when receiving this query on (m, ID, t) under the public key $PK_{ID} = (R_{ID}, P_{ID})$, the challenge $\mathcal{C}$ performs the following steps to generate a valid signature. First, $\mathcal{C}$ obtains the corresponding tuple $(m, ID, PK_{ID}, t, v, T_2, coin)$ from L_2. Then, we discuss two cases.
  1. If $ID = ID'$ and $coin = 1$, $\mathcal{C}$ returns failure and terminates.
  2. Otherwise, $\mathcal{C}$ chooses three random values $h, u, w \in \mathbb{Z}_q^*$ and sets the signature $\sigma = w \cdot P_{ID} + v \cdot R_{ID} + v \cdot h \cdot P_{pub} + u \cdot P_{pub}$. Then, $\mathcal{C}$ adds (ID, d_{ID}, R_{ID}, h) in K^{list}, $(ID, t, u, T_0, T_{ID,t})$ in L_0, and $(m, ID, PK_{ID}, t, w, T_1)$ in L_1, respectively. Finally, $\mathcal{C}$ responds with the signature σ. Here, we should emphasize that, even though $\mathcal{C}$ does not hold the corresponding secret value, initial key and time update key of the user with identity ID, the signature σ is still valid and can satisfy the verification equation by

$$\begin{aligned}\hat{e}(P,\sigma) &= \hat{e}(P, w\cdot P_{ID} + v\cdot R_{ID} + v\cdot h\cdot P_{pub} + u\cdot P_{pub})\\ &= \hat{e}(P, w\cdot P_{ID})\cdot\hat{e}(P, v\cdot R_{ID} + v\cdot h\cdot P_{pub})\cdot\hat{e}(P, u\cdot aP)\\ &= \hat{e}(w\cdot P, P_{ID})\cdot\hat{e}(v\cdot P, R_{ID} + h\cdot P_{pub})\cdot\hat{e}(aP, u\cdot P)\\ &= \hat{e}(P_{ID}, T_1)\cdot\hat{e}(R_{ID} + h\cdot P_{pub}, T_2)\cdot\hat{e}(P_{pub}, T_0).\end{aligned}$$

– *Forgery*. Assume that the adversary $\mathcal{A}$ can forge a valid signature tuple $(m^*, ID^*, t^*, \sigma^*)$ under the public key PK_{ID^*}, where PK_{ID^*} is the original public key without being replaced. First, the challenge $\mathcal{C}$ obtains the corresponding tuple $(m, ID, PK_{ID}, t, v, T_2, coin)$ from L_2. We discuss three cases.
1. If $ID^* \neq ID'$, $\mathcal{C}$ returns failure and terminates.
2. If $ID^* = ID'$ and $coin = 0$, $\mathcal{C}$ returns failure and terminates.
3. If $ID^* = ID'$ and $coin = 1$, $\mathcal{C}$ can employ the forgery signature σ^* to solve the CDH problem as follows. Since σ^* is valid, the equality

$$\hat{e}\big(P,\sigma^*\big) = \hat{e}(P_{ID^*}, T_1)\cdot\hat{e}(R_{ID^*} + h\cdot P_{pub}, T_2)\cdot\hat{e}(P_{pub}, T_0).$$

holds. Then, by the property of bilinear pairings, the last equality becomes

$$\begin{aligned}\hat{e}(P,\sigma^*) &= \hat{e}(P_{ID^*}, T_1)\cdot\hat{e}(R_{ID^*} + h\cdot P_{pub}, T_2)\cdot\hat{e}(P_{pub}, T_0)\\ &= \hat{e}(P_{ID^*}, w\cdot P)\cdot\hat{e}(r_{ID^*}\cdot P + h\cdot aP, v\cdot bP)\cdot\hat{e}(aP, u\cdot P)\\ &= \hat{e}(w\cdot P_{ID^*}, P)\cdot\hat{e}(r_{ID^*}\cdot P, v\cdot bP)\cdot\hat{e}(h\cdot aP, v\cdot bP)\cdot\hat{e}(aP, u\cdot P)\\ &= \hat{e}(w\cdot P_{ID^*}, P)\cdot\hat{e}(r_{ID^*}\cdot v\cdot bP, P)\cdot\hat{e}(h\cdot v\cdot abP, P)\cdot\hat{e}(u\cdot aP, P)\\ &= \hat{e}(P, w\cdot P_{ID^*} + r_{ID^*}\cdot T_2 + h\cdot v\cdot abP + u\cdot P_{pub}),\end{aligned}$$

which yields $\sigma^* = w\cdot P_{ID^*} + r_{ID^*}\cdot T_2 + h\cdot v\cdot abP + u\cdot P_{pub}$. Hence, $\mathcal{C}$ can obtain the solution abP of the CDH problem by computing $(\sigma^* - w\cdot P_{ID^*} - r_{ID^*}\cdot T_2 - u\cdot P_{pub})/h\cdot v$.

By the responses to the oracle queries mentioned above, $\mathcal{C}$ has perfectly simulated the challenger in Game I. Next, we evaluate the advantage that the challenger $\mathcal{C}$ wins the game by obtaining the solution of the CDH problem. In order to simplify the analysis, we define the events as follows:

A: $\mathcal{C}$ does not abort in the Query phase.
B: The forged signature σ^* is valid.
C: $\mathcal{C}$ does not abort in the Forgery phase.

By above, the challenger $\mathcal{C}$ does not abort in the *Initial key extract* query with the probability $(1 - 1/q_e)$, and does not abort in the *Super sign* query with the probability $(1-\delta/q_e)^{q_{ss}}$. Hence, $\Pr[A] = (1-1/q_e)(1-\delta/q_e)^{q_{ss}} \geqslant (1-1/q_e)(1-\delta)^{q_{ss}}$. In addition, $\Pr[B] = \epsilon$ and $\Pr[C] = \delta/q_e$. Since A, B and C are independent, we obtain the probability

$$\Pr[A\wedge B\wedge C] = \Pr[A]\cdot\Pr[B]\cdot\Pr[C] \geqslant (1-1/q_e)(1-\delta)^{q_{ss}}(\delta/q_t)\epsilon.$$

Meanwhile, note that $(1-\delta)^{q_{ss}}\delta/q_e$ is maximized at $\delta_{opt}=1/(1+q_{ss})$. Hence, $\mathcal{C}$ has an advantage

$$\epsilon' \geqslant (1-1/q_e)\big(1-1/(1+q_{ss})\big)^{q_{ss}}\big(1/q_e(1+q_{ss})\big)\epsilon.$$

Finally, by the required computation time for answering queries in the simulation game above, we conclude with $\tau'=\tau+O(q_0+q_1+q_2+q_e+q_{sv}+q_{ss})\tau_1$, where τ_1 is the time to perform a scalar multiplication in $\mathbb{G}$. □

Theorem 2. *The proposed RCLSS scheme is provably secure against the UF-RCLSS-ACMA attack of Type II adversary (honest-but-curious KGC). Concretely, assume that an UF-RCLSS-ACMA adversary $\mathcal{A}$ with a non-negligible advantage ϵ can break the proposed RCLSS scheme within running time. Then, we can construct an algorithm $\mathcal{C}$ that can solve the CDH problem with a non-negligible advantage*

$$\epsilon' \geqslant (1-1/q_{sv})\big(1-1/(1+q_{ss})\big)^{q_{ss}}\big(1/q_{sv}(1+q_{ss})\big)\epsilon$$

within running time $\tau'=\tau+O(q_0+q_1+q_2+q_{sv}+q_{ss})\tau_1$, where, in running time τ', $\mathcal{A}$ may ask at most q_i queries to the random oracles H_i $(i=0,1,2)$, q_{sv} queries to the Secret value extract *oracle and q_{ss} queries to the* Super sign *oracle, respectively; τ_1 is the time needed to perform a scalar multiplication in $\mathbb{G}$.*

Proof. Given a random instance (P, aP, bP) of the CDH problem in $\mathbb{G}$, where P, aP, $bP \in \mathbb{G}$ with unknown a, $b \in \mathbb{Z}_q^*$, we would like to construct an algorithm $\mathcal{C}$ to output the CDH solution abP. Here, the algorithm $\mathcal{C}$ will play the challenger in Game II and interacts with the adversary $\mathcal{A}$ as follows.

– *Setup.* The challenger $\mathcal{C}$ sets $P_{pub}=sP$, where s is master secret key. Then, the challenger $\mathcal{C}$ sets public parameters $PP=\langle\mathbb{G},\mathbb{G}_T,q,\hat{e},P,P_{pub},f,H_0,H_1,H_2\rangle$, where the hash functions f, H_0, H_1 and H_2 are random oracles controlled by $\mathcal{C}$. Finally, $\mathcal{C}$ returns the master secret key s and the public parameters PP to $\mathcal{A}$. Since $\mathcal{A}$ knows the master secret key s, it can compute the initial key d_{ID}, the first partial public key R_{ID} and time update key $T_{ID,t}$ so that it does not need to issue the *Initial key extract* and *Time update key* queries.

– *Queries.* $\mathcal{C}$ first chooses a random $i\in[1,q_{sv}]$ and sets ID' as the target identity of the i-th query to the *Set secret value* oracle. $\mathcal{C}$ answers the queries issued by $\mathcal{A}$ as follows.

- f *queries*: see the proof of Theorem 1.
- H_0 *queries*: see the proof of Theorem 1.
- H_1 *queries*: at any time, $\mathcal{A}$ can issue queries with (m, ID, PK_{ID}, t) to the random oracle H_1. To respond these queries, $\mathcal{C}$ maintains an initially-empty list L_1 of tuples of the form $\langle m, ID, PK_{ID}, t, w, T_1, coin>$ and responds as follows.
  1. If (m, ID, PK_{ID}, t) already appears in L_1 with a tuple $\langle m, ID, PK_{ID}, t, w, T_1, coin\rangle$, then $\mathcal{C}$ responds with T_1.

2. Otherwise, we split into two cases. If $ID \neq ID'$, $\mathcal{C}$ randomly chooses $w \in \mathbb{Z}_q^*$, and sets $T_1 = w \cdot P$ and $coin = 0$. If $ID = ID'$, $\mathcal{C}$ first flips a random $coin \in \{0, 1\}$ with $\Pr[coin = 1] = \delta$ for some δ that will be determined later. $\mathcal{C}$ sets $T_1 = w \cdot P$ if $coin = 0$ and $T_1 = w \cdot bP$ if $coin = 1$. Finally, $\mathcal{C}$ adds $\langle m, ID, PK_{ID}, t, w, T_1, coin\rangle$ in L_1 and responds with T_1.

- *H_2 queries*: at any time, $\mathcal{A}$ can issue queries with (m, ID, PK_{ID}, t) to the random oracle H_2. To respond these queries, $\mathcal{C}$ maintains an initially-empty list L_2 of tuples of the form $\langle m, ID, PK_{ID}, t, v, T_2\rangle$ and responds as follows.
  1. If (m, ID, PK_{ID}, t) already appears in L_2 with a tuple $\langle m, ID, PK_{ID}, t, v, T_2\rangle$, then $\mathcal{C}$ responds with T_2.
  2. Otherwise, $\mathcal{C}$ randomly chooses $v \in \mathbb{Z}_q^*$, sets $T_2 = v \cdot P$, adds $\langle m, ID, PK_{ID}, t, v, T_2\rangle$ in L_2 and responds with T_2.
- *Set secret value queries*: upon receiving this query with ID, $\mathcal{C}$ maintains a list P^{list} of tuples of the form $\langle ID, x_{ID}, P_{ID}\rangle$ and responds as follows.
  1. If ID already appears in the list P^{list} with a tuple $\langle ID, x_{ID}, P_{ID}\rangle$, then $\mathcal{C}$ responds with the secret value x_{ID} and the second partial public key P_{ID}.
  2. Otherwise, we split into two cases. If $ID = ID'$, $\mathcal{C}$ returns failure and terminates. If $ID \neq ID'$, $\mathcal{C}$ randomly chooses $x_{ID} \in \mathbb{Z}_q^*$, computes $P_{ID} = x_{ID} \cdot P$, adds the tuple $\langle ID, x_{ID}, P_{ID}\rangle$ in P^{list}, and responds with x_{ID} and P_{ID}.
- *Public key retrieve queries*: upon receiving this query with ID, $\mathcal{C}$ maintains a list PK^{list} of tuples of the form $\langle ID, x_{ID}, P_{ID}, R_{ID}\rangle$ and responds as follows.
  1. If ID already appears in PK^{list} with a tuple $\langle ID, x_{ID}, P_{ID}, R_{ID}\rangle$, then $\mathcal{C}$ responds with the public key $PK_{ID} = (R_{ID}, P_{ID})$.
  2. Otherwise, $\mathcal{C}$ issues *Set secret value* queries to obtain P_{ID}, and adds the tuple $\langle ID, x_{ID}, P_{ID}, R_{ID}\rangle$ in PK^{list}. Then, $\mathcal{C}$ responds with the public key $PK_{ID} = (R_{ID}, P_{ID})$. Note that if $ID = ID'$, $\mathcal{C}$ sets $P_{ID} = aP$ and $x_{ID} = \perp$, and stores the tuple (ID, x_{ID}, P_{ID}) in P^{list} in advance when setting ID as the target identity before performing all queries.
- *Public key replace queries*: see the proof of Theorem 1.
- *Signing queries*: when receiving this query on (m, ID, t) under the public key $PK_{ID} = (R_{ID}, P_{ID})$, the challenge $\mathcal{C}$ performs the following steps to generate a valid signature. First, $\mathcal{C}$ obtains the corresponding tuple $(m, ID, PK_{ID}, t, w, T_1, coin)$ from L_1. Then, we discuss two cases.
  1. If $ID = ID'$ and $coin = 1$, $\mathcal{C}$ returns failure and terminates.
  2. Otherwise, $\mathcal{C}$ chooses three random values h, u, $v \in \mathbb{Z}_q^*$ and sets the signature $\sigma = w \cdot P_{ID} + v \cdot R_{ID} + v \cdot h \cdot P_{pub} + u \cdot P_{pub}$. Then, $\mathcal{C}$ adds (ID, d_{ID}, R_{ID}, h) in L_f, $(ID, t, u, T_0, T_{ID,t})$ in L_0, and $(m, ID, PK_{ID}, t, v, T_2)$ in L_2, respectively, and returns the signature σ. Finally, as in the proof of Theorem 1, the signature σ is valid and can pass the verification.

– *Forgery*. Assume that the adversary $\mathcal{A}$ can forge a valid signature tuple $(m^*, ID^*, t^*, \sigma^*)$ under the public key PK_{ID^*}, where PK_{ID^*} is the original public key without being replaced. First, the challenge $\mathcal{C}$ obtains the corresponding tuple $(m, ID, PK_{ID}, t, v, T_1, coin)$ from L_1. We discuss three cases.

1. If $ID^* \neq ID'$, $\mathcal{C}$ returns failure and terminates.
2. If $ID^* = ID'$ and $coin = 0$, $\mathcal{C}$ returns failure and terminates.
3. If $ID^* = ID'$ and $coin = 1$, $\mathcal{C}$ can employ the forgery signature σ^* to solve the CDH problem. Since σ^* is valid, the equality

$$\hat{e}\big(P, \sigma^*\big) = \hat{e}(P_{ID^*}, T_1) \cdot \hat{e}(R_{ID^*} + h \cdot P_{pub}, T_2) \cdot \hat{e}(P_{pub}, T_0)$$

holds. Then, by the property of bilinear pairings, the last equality becomes

$$\begin{aligned} \hat{e}\big(P, \sigma^*\big) &= \hat{e}(P_{ID^*}, T_1) \cdot \hat{e}(R_{ID^*} + h \cdot P_{pub}, T_2) \cdot \hat{e}(P_{pub}, T_0) \\ &= \hat{e}(aP, w \cdot bP) \cdot \hat{e}(r_{ID^*} \cdot P + h \cdot s \cdot P, v \cdot P) \cdot \hat{e}(s \cdot P, u \cdot P) \\ &= \hat{e}(w \cdot abP, P) \cdot \hat{e}(R_{ID^*}, v \cdot P) \cdot \hat{e}(P, u \cdot s \cdot P) \\ &= \hat{e}(P, w \cdot abP + v \cdot R_{ID^*} + u \cdot s \cdot P), \end{aligned}$$

which yields $\sigma^* = w \cdot abP + v \cdot R_{ID^*} + u \cdot s \cdot P$. Hence, $\mathcal{C}$ can obtain the solution abP of the CDH problem by computing $(\sigma^* - v \cdot R_{ID^*} - u \cdot s \cdot P)/w$.

According to the responses to the oracle queries mentioned above, $\mathcal{C}$ has perfectly simulated the challenger in Game II. Next, we evaluate the advantage of $\mathcal{C}$ in winning the game by obtaining the solution of the CDH problem. In order to simplify the analysis, we define the events as follows:

A: $\mathcal{C}$ does not abort in the Query phase.
B: The forged signature σ^* is valid.
C: $\mathcal{C}$ does not abort in the Forgery phase.

By above, the challenger $\mathcal{C}$ does not abort in the *Set secret value* query with the probability $(1 - 1/q_{sv})$, and does not abort in the *Super sign* query with the probability $(1 - \delta/q_{sv})^{q_{ss}}$. Hence, $\Pr[A] = (1 - 1/q_{sv})(1 - \delta/q_{sv})^{q_{ss}} \geqslant (1 - 1/q_{sv})(1 - \delta)^{q_{ss}}$. In addition, $\Pr[B] = \epsilon$ and $\Pr[C] = \delta/q_{sv}$. Since A, B and C are independent, we obtain the probability

$$\Pr[A \wedge B \wedge C] = \Pr[A] \cdot \Pr[B] \cdot \Pr[C] \geqslant (1 - 1/q_{sv})(1 - \delta)^{q_{ss}}(\delta/q_{sv})\epsilon.$$

Meanwhile, note that $(1 - \delta)^{q_{ss}}(\delta/q_{sv})$ is maximized at $\delta_{opt} = 1/(1 + q_{ss})$. Hence, $\mathcal{C}$ has an advantage

$$\epsilon' \geqslant (1 - 1/q_{sv})\big(1 - 1/(1 + q_{ss})\big)^{q_{ss}}\big(1/q_{sv}(1 + q_{ss})\big)\epsilon.$$

Finally, by the required computation time for answering queries in the simulation game above, we conclude with $\tau' = \tau + O(q_0 + q_1 + q_2 + q_{sv} + q_{ss})\tau_1$, where τ_1 is the time to perform a scalar multiplication in $\mathbb{G}$. □

Theorem 3. *The proposed RCLSS scheme is provably secure against the UF-RCLSS-ACMA attack of Type III adversary (revoked user). Concretely, assume that an UF-RCLSS-ACMA adversary $\mathcal{A}$ with a non-negligible advantage ϵ can break the proposed RCLSS* *scheme within running time τ. Then, we can construct an algorithm $\mathcal{C}$ that can solve the CDH problem with a non-negligible advantage*

$$\epsilon' \geqslant (1 - 1/q_t)\big(1 - 1/(1 + q_{ss})\big)^{q_{ss}}\big(1/q_t(1 + q_{ss})\big)\epsilon$$

within running time $\tau' = \tau + O(q_0 + q_1 + q_2 + q_e + q_{sv} + q_{ss})\tau_1$, where, in running time τ', $\mathcal{A}$ may ask at most q_i queries to the random oracles H_i $(i = 0, 1, 2)$, q_e queries to the Initial key extract *oracle, q_t queries to the* Time key update *oracle, q_{sv} queries to the* Secret value extract *oracle and q_{ss} queries to the* Super sign *oracle, respectively*; τ_1 *is the time needed to perform a scalar multiplication in $\mathbb{G}$.*

Proof. Given a random instance (P, aP, bP) of the CDH problem in $\mathbb{G}$, where P, aP, $bP \in \mathbb{G}$ with unknown a, $b \in \mathbb{Z}_q^*$, we would like to construct an algorithm $\mathcal{C}$ to output the CDH solution abP. Here, the algorithm $\mathcal{C}$ will play the challenger in Game III and interacts with the adversary $\mathcal{A}$ as follows.

– *Setup*. The challenger $\mathcal{C}$ sets $P_{pub} = aP$ and the public parameters $PP = \langle \mathbb{G}, \mathbb{G}_T, q, \hat{e}, P, P_{pub}, f, H_0, H_1, H_2 \rangle$, where the hash functions f, H_0, H_1 and H_2 are random oracles controlled by $\mathcal{C}$. Finally, $\mathcal{C}$ returns PP to $\mathcal{A}$.

– *Queries*. $\mathcal{C}$ first chooses a random $i \in [1, q_t]$ and sets ID' as the target identity of the i-th query to the *Time key update* oracle. $\mathcal{C}$ answers the queries issued by $\mathcal{A}$ as follows.

- f *queries*: see the proof of Theorem 1.
- H_0 *queries*: at any time, $\mathcal{A}$ can issue queries with (ID, t) to the random oracle H_0. To respond these queries, $\mathcal{C}$ maintains an initially-empty list L_0 of tuples of the form $\langle ID, t, u, T_0, T_{ID,t}, coin \rangle$ and responds as follows.
  1. If (ID, t) already appears in L_0 with a tuple $\langle ID, t, u, T_0, T_{ID,t}, coin \rangle$, then $\mathcal{C}$ responds with T_0.
  2. Otherwise, we split into two cases. If $ID \neq ID'$, $\mathcal{C}$ randomly chooses $u \in \mathbb{Z}_q^*$, and sets $T_0 = u \cdot P$, $T_{ID,t} = u \cdot P_{pub}$, and $coin = 0$. If $ID = ID'$, $\mathcal{C}$ first flips a random $coin \in \{0, 1\}$ with $\Pr[coin = 1] = \delta$ for some δ that will be determined later. $\mathcal{C}$ sets $T_0 = u \cdot P$ and $T_{ID,t} = u \cdot P_{pub}$ if $coin = 0$, and $T_0 = u \cdot bP$ and $T_{ID,t} = \perp$ if $coin = 1$. Finally, $\mathcal{C}$ adds $\langle ID, t, u, T_0, T_{ID,t} coin \rangle$ in L_0 and responds with T_0.
- H_1 *queries*: see the proof of Theorem 1.
- H_2 *queries*: see the proof of Theorem 2.
- *Initial key extract queries*: upon receiving this query with ID, $\mathcal{C}$ maintains a list K^{list} of tuples of the form $\langle ID, d_{ID}, R_{ID}, h \rangle$ and responds as follows.
  1. If ID already appears in K^{list} with a tuple $\langle ID, d_{ID}, R_{ID}, h \rangle$, then $\mathcal{C}$ responds with the initial key d_{ID} and the first partial public key R_{ID}.
  2. Otherwise, $\mathcal{C}$ randomly chooses h, $z \in \mathbb{Z}_q^*$, and sets $R_{ID} = z \cdot P - h \cdot P_{pub}$, $d_{ID} = z$ with $f(ID, R_{ID}) = h$. Moreover, if this tuple $\langle ID, R_{ID}, h \rangle$ already exists in L_f, $\mathcal{C}$ needs to randomly pick $h, z \in \mathbb{Z}_q^*$ repeatedly until the tuple is new. Finally, $\mathcal{C}$ adds the tuple $\langle ID, d_{ID}, R_{ID}, h \rangle$ in K^{list} and responds with d_{ID} and R_{ID}.

- *Set secret value queries*: see the proof of Theorem 1.
- *Public key retrieve queries*: upon receiving this query with ID, $\mathcal{C}$ maintains a list PK^{list} of tuples of the form $\langle ID, x_{ID}, R_{ID}, P_{ID}\rangle$ and responds as follows.
  1. If ID already appears in PK^{list} with a tuple $\langle ID, x_{ID}, R_{ID}, P_{ID}\rangle$, then $\mathcal{C}$ responds with the public key $PK_{ID} = (R_{ID}, P_{ID})$.
  2. Otherwise, $\mathcal{C}$ issues the *Initial key extract* and *Set secret value* queries to obtain R_{ID} and P_{ID}, and adds the tuple $\langle ID, x_{ID}, R_{ID}, P_{ID}\rangle$ in PK^{list}. Then, $\mathcal{C}$ responds with the public key $PK_{ID} = (R_{ID}, P_{ID})$.
- *Public key replace queries*: see the proof of Theorem 1.
- *Time key update queries*: upon receiving this query with (ID, t), $\mathcal{C}$ looks up the corresponding tuple $\langle ID, t, u, T_0, T_{ID,t}, coin\rangle$ in L_0. We discuss two cases.
  1. If the tuple appears in L_0, then $\mathcal{C}$ responds with the time update key $T_{ID,t}$.
  2. Otherwise, we split into two cases. If $ID = ID'$, the challenger $\mathcal{C}$ returns failure and terminates. If $ID \neq ID'$, $\mathcal{C}$ issues the H_0 query to obtain $T_{ID,t}$ and returns it to $\mathcal{A}$.
- *Signing queries*: when receiving this query on (m, ID, t) under the public key $PK_{ID} = (R_{ID}, P_{ID})$, the challenge $\mathcal{C}$ performs the following steps to generate a valid signature. First, $\mathcal{C}$ obtains the corresponding tuple $(ID, t, u, T_0, T_{ID,t}, coin)$ from L_0. Then, we discuss two cases.
  1. If $ID = ID'$ and $coin = 1$, $\mathcal{C}$ returns failure and terminates.
  2. If $ID = ID'$ and $coin = 0$, $\mathcal{C}$ chooses three random values $h, v, w \in \mathbb{Z}_q^*$ and sets the signature $\sigma = w \cdot P_{ID} + v \cdot R_{ID} + v \cdot h \cdot P_{pub} + u \cdot P_{pub}$. Then, $\mathcal{C}$ adds (ID, d_{ID}, R_{ID}, h) in K^{list}, $(m, ID, PK_{ID}, t, w, T_1)$ in L_1 and $(m, ID, PK_{ID}, t, v, T_2)$ in L_2, respectively, and return the signature σ. Finally, as in the proof of Theorem 1, the signature σ is valid and can pass the verification.

– *Forgery*. Assume that the adversary $\mathcal{A}$ can forge a valid signature tuple $(m^*, ID^*, t^*, \sigma^*)$ under the public key PK_{ID^*}, where PK_{ID^*} is the original public key without being replaced. First, the challenge $\mathcal{C}$ obtains the corresponding tuple $(ID, t, u, T_0, T_{ID,t}, coin)$ from L_0. We discuss three cases.
  1. If $ID^* \neq ID'$, $\mathcal{C}$ returns failure and terminates.
  2. If $ID^* = ID'$ and $coin = 0$, $\mathcal{C}$ returns failure and terminates.
  3. If $ID^* = ID'$ and $coin = 1$, $\mathcal{C}$ can employ the forgery signature σ^* to solve the CDH problem. Since σ^* is valid, the equality

$$\hat{e}(P, \sigma^*) = \hat{e}(P_{ID^*}, T_1) \cdot \hat{e}(R_{ID^*} + h \cdot P_{pub}, T_2) \cdot \hat{e}(P_{pub}, T_0)$$

holds. Then, by the property of bilinear pairings, the last equality becomes

$$\begin{aligned}
\hat{e}(P, \sigma^*) &= \hat{e}(P_{ID^*}, T_1) \cdot \hat{e}(R_{ID^*} + h \cdot P_{pub}, T_2) \cdot \hat{e}(P_{pub}, T_0) \\
&= \hat{e}(P_{ID^*}, w \cdot P) \cdot \hat{e}(r_{ID^*} \cdot P + h \cdot aP, v \cdot P) \cdot \hat{e}(aP, u \cdot bP) \\
&= \hat{e}(w \cdot P_{ID^*}, P) \cdot \hat{e}(r_{ID^*} \cdot P, v \cdot P) \cdot \hat{e}(h \cdot aP, v \cdot P) \cdot \hat{e}(aP, u \cdot bP)
\end{aligned}$$

$$= \hat{e}(w \cdot P_{ID^*}, P) \cdot \hat{e}(v \cdot r_{ID^*} \cdot P, P) \cdot \hat{e}(h \cdot v \cdot aP, P) \cdot \hat{e}(u \cdot abP, P)$$
$$= \hat{e}(P, w \cdot P_{ID^*} + v \cdot R_{ID^*} + h \cdot v \cdot P_{pub} + u \cdot abP),$$

which yields $\sigma^* = w \cdot P_{ID^*} + v \cdot R_{ID^*} + h \cdot v \cdot P_{pub} + u \cdot abP$. Hence, $\mathcal{C}$ can obtain the solution abP of the CDH problem by computing $(\sigma^* - w \cdot P_{ID^*} - v \cdot R_{ID^*} - h \cdot v \cdot P_{pub})/u$.

By the responses to the oracle queries mentioned above, $\mathcal{C}$ has perfectly simulated the challenger in Game III. Next, we evaluate the advantage that the challenger $\mathcal{C}$ wins the game by obtaining the solution of the CDH problem. In order to simplify the analysis, we define the events as follows:

A: $\mathcal{C}$ does not abort in the Query phase.
B: The forged signature σ^* is valid.
C: $\mathcal{C}$ does not abort in the Forgery phase.

By above, the challenger $\mathcal{C}$ does not abort in the *Time update key* query with the probability $(1 - 1/q_t)$, and does not abort in the *Super sign* query with the probability $(1 - \delta/q_t)^{q_{ss}}$. Hence, $\Pr[A] = (1 - 1/q_t)(1 - \delta/q_t)^{q_{ss}} \geqslant (1 - 1/q_t)(1 - \delta)^{q_{ss}}$. In addition, $\Pr[B] = \epsilon$ and $\Pr[C] = \delta/q_t$. Since A, B and C are independent, we obtain the probability

$$\Pr[A \wedge B \wedge C] = \Pr[A] \cdot \Pr[B] \cdot \Pr[C] \geqslant (1 - 1/q_t)(1 - \delta)^{q_{ss}}(\delta/q_t)\epsilon.$$

Meanwhile, note that $(1 - \delta)^{q_{ss}}\delta/q_t$ is maximized at $\delta_{opt} = 1/(1 + q_{ss})$. Hence, $\mathcal{C}$ has an advantage

$$\epsilon' \geqslant (1 - 1/q_t)\big(1 - 1/(1 + q_{ss})\big)^{q_{ss}}\big(1/q_t(1 + q_{ss})\big)\epsilon.$$

Finally, by the required computation time for answering queries in the simulation game above, we conclude with $\tau' = \tau + O(q_0 + q_1 + q_2 + q_e + q_{sv} + q_{ss})\tau_1$, where τ_1 is the time to perform a scalar multiplication in $\mathbb{G}$. □

6. Comparisons

Here, we compare our proposed RCLSS scheme with Tso *et al.*'s CLSS scheme (2012) and Sun *et al.*'s RCLS scheme (2014). For computational comparisons, we define several notations of time-consuming operations as follows.

- T_p: the time of executing a bilinear pairing operation $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$;
- T_m: the time of executing a scalar multiplication in $\mathbb{G}$, an exponentiation in $\mathbb{G}_T$ or an exponentiation operation in $\mathbb{Z}_q^*$;
- T_h: the time of executing a map-to-point hash function.

For signature sizes, a popular and valid choice of bilinear pairings is the Ate pairing system in Scott *et al.* (2006) which adopts an elliptic curve over a finite field $E(Fp)$, with $p = 512$ bits.

Table 1
Comparisons between our scheme and two previously proposed schemes.

	Tso *et al.*'s CLSS scheme (2012)	Sun *et al.*'s RCLS scheme (2014)	Our RCLSS scheme
Signature size	1024 bits	2048 bits	1024 bits
Computational cost of signing	$2T_m + T_h$	$3T_m + 2T_h$	$2T_m + 2T_h$
Computational cost of verification	$3T_p + T_m + T_h$	$4T_p + 3T_h$	$4T_p + T_m + 3T_h$
Required channel for revocation	Secure channel	Public channel	Public channel
Against Type I adversary	No (Du and Wen, 2014)	Yes	Yes
Against Type II adversary	Yes	Yes	Yes
Against Type III adversary	Not required	Yes	Yes

Table 1 below lists the comparisons between our RCLSS scheme and the schemes in Tso *et al.* (2012) and Sun *et al.* (2014), in terms of signature sizes, computational costs of signing and verification, revocable functionality and security properties. As for the signature size, it is clear that our scheme requires only one element in $\mathbb{G}$. On the other hand, for the computational costs of the sign phase, our scheme performs better than Sun *et al.*'s scheme but requires one more hash function operation than Tso *et al.*'s scheme. For the computational costs of the verification phase, Tso *et al.*'s scheme requires the least computation but their scheme has been shown insecure against the Type I adversary (Du and Wen, 2014). The revocation mechanism of Tso *et al.*'s scheme requires a secure channel to transmit new initial keys periodically to non-revoked users, which causes enormous computation workload for the KGC and users. According to Table 1, our scheme possesses the shortest signature size, provable security and the revocation mechanism with public channels while retaining efficiency.

7. Extended Application

A user authentication scheme is a mechanism which enables a server to authenticate a remote user (client) over an open network. Nowadays, clients use mobile devices to access multiple servers for various services. Before doing so, clients should be authenticated by servers. In this section, we extend our RCLSS scheme to construct a cloud-revocation-authority (CRA)-aided authentication scheme with period-limited privileges for mobile multi-server environments. In this CRA-aided authentication scheme, one or multiple cloud revocation authorities (CRAs) replace the role of the KGC to be responsible for the revocation. To provide various privileges, time update keys used in our RCLSS scheme are replaced with multiple period-limited privilege keys. A CRA with a master privilege key is responsible to issue period-limited privilege keys for users to access some service server at various periods. In the CRA-aided authentication scheme with period-limited privileges, there are k independent CRAs responsible for managing k independent service servers, respectively. The KGC randomly selects k different master privilege keys $\beta_1, \beta_2, \dots, \beta_k$ and sends each β_j to the corresponding CRA_j, respectively. Under some circumstance, if the KGC wants to share all the revocation responsibility with a designated

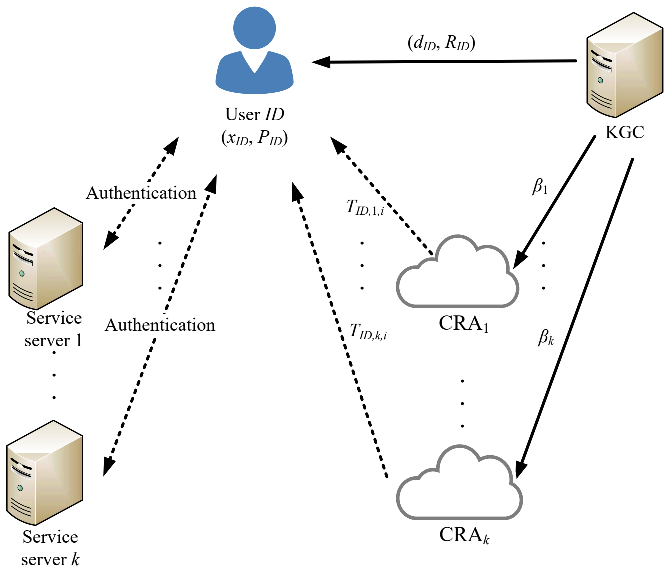

Fig. 1. System model of CRA-aided authentication scheme.

CRA, all the master privilege keys will be sent to this CRA so that it can simultaneously manage all the k service servers. Also, the KGC sends the initial key pair (d_{ID}, R_{ID}) to a legitimate user with identity ID via a secure channel. On the other hand, if this user with identity ID is granted an access to the service server j at period i, the CRA_j will use the master privilege key β_j to generate the period-limited privilege key $T_{ID,j,i}$ and send it to the user via a public channel. The full private key of the user is the combination of the initial key pair (d_{ID}, R_{ID}), the period-limited privilege key $T_{ID,j,i}$, and the secret value x_{ID} chosen by the user herself/himself. Consequently, a user can access the server j at period i provided that the signature generated by using the user's full private key is valid. The system model of CRA-aided authentication scheme is depicted in Fig. 1.

7.1. *CRA-Aided Authentication Scheme with Period-Limited Privileges*

The proposed CRA-aided authentication scheme with period-limited privileges consists of eight algorithms:

– *System setup*: as the System setup phase in the RCLSS scheme proposed in Section 4, a trusted KGC generates a master secret key α and computes the system public key $P_{pub} = \alpha \cdot P$. In addition, suppose that there are k independent service servers managed by k independent CRAs in the system. The KGC randomly selects k different master privilege keys $\beta_1, \beta_2, \dots, \beta_k$ and sends each β_j to the corresponding CRA_j via a secure channel, respectively. In the meantime, the KGC also computes the privilege public key $C_{pub,j} = \beta_j \cdot P$ for each CRA_j. The KGC selects four hash functions $f\colon \{0,1\}^* \to \mathbb{Z}_q^*$, and H_0,

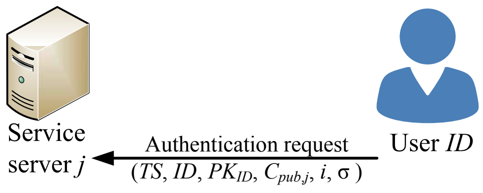

Fig. 2. Authentication procedure.

H_1, H_2: $\{0,1\}^* \to \mathbb{G}$. Finally, the PKG publishes the public parameters $PP = \rangle\mathbb{G}, \mathbb{G}_T, q, \hat{e}, P, P_{pub}, C_{pub,1}, C_{pub,2}, \dots, C_{pub,k}, f, H_0, H_1, H_2\rangle$.

– *Initial key extract*: the KGC sends the initial key d_{ID} and the first partial public key R_{ID} to the user via a secure channel.
– *Privilege key extract*: suppose that a user with identity $ID \in \{0,1\}^*$ is granted an access to the service server j at period i. The corresponding CRA_j uses the master privilege key β_j to generate the period-limited privilege key $T_{ID,j,i} = \beta_j \cdot H_0(ID, i)$ and send it to the user via a public channel.
– *Set secret value*: the user with identity ID randomly selects a secret value $x_{ID} \in \mathbb{Z}_q^*$ and computes the second partial public key $P_{ID} = x_{ID} \cdot P$.
– *Set private key*: the user with identity ID sets her/his full private key $sk_{ID,j,i} = (d_{ID}, T_{ID,j,i}, x_{ID})$ for the period i.
– *Set public key*: a user with R_{ID} and P_{ID} sets her/his public key $PK_{ID} = (R_{ID}, P_{ID})$.
– *Authentication*: if a user with identity ID wants to access some service server j at period i, the user sends an authentication request along with ID and period i to the service server j. The authentication procedure is depicted in Fig. 2 below and the detailed steps are presented as follows.
  • For a current time stamp TS, a user first computes $T_1 = H_1(TS, ID, PK_{ID}, P_{pub}, C_{pub,j}, i)$, and $T_2 = H_2(TS, ID, PK_{ID}, P_{pub}, C_{pub,j}, i)$. The user then uses the private key $sk_{ID,t} = (d_{ID}, T_{ID,j,i}, x_{ID})$ to generate the signature $\sigma = x_{ID} \cdot T_1 + d_{ID} \cdot T_2 + T_{ID,j,i}$, and sends an authentication request with $(TS, ID, PK_{ID}, C_{pub,j}, i, \sigma)$ to the service server j.
  • Upon receiving an authentication request with $(TS, ID, PK_{ID}, C_{pub,j}, i, \sigma)$, the service server j first validates the correctness of the current time stamp TS. If not valid, output "reject". Otherwise, the service server j computes $h = f(ID, R_{ID})$, $T_0 = H_0(ID, i)$, $T_1 = H_1(TS, ID, PK_{ID}, P_{pub}, C_{pub,j}, i)$ and $T_2 = H_2(TS, ID, PK_{ID}, P_{pub}, C_{pub,j}, i)$, and then verifies the equality $\hat{e}(P, \sigma) = \hat{e}(P_{ID}, T_1) \cdot \hat{e}(R_{ID} + h \cdot P_{pub}, T_2) \cdot \hat{e}(C_{pub,j}, T_0)$. If it holds, output "accept". Otherwise, output "reject".

7.2. *Discussions*

There is no doubt that authentication schemes may be implemented by signature schemes. In our CRA-aided authentication scheme, a service server verifies a signature generated with user's full private key at a given period. Our CRA-aided authentication scheme aims at user identification and authorization before accessing service servers. Note that the current time stamp TS may be replaced with a challenge message sent by the service server j. Although our scheme does not concern the construction of mutual authentication and session key establishment, some existing session key exchange or SSL protocols (Tseng *et al.*, 2008; Chuang and Tseng, 2012; Kaufman *et al.*, 2014; Freier *et al.*, 2011) can be employed to establish a secure session key for providing communication confidentiality. In the following, based on the strong unforgeability security of the RCLSS scheme, we prove that the proposed CRA-aided authentication scheme is provable secure under active attacks.

Theorem 4. *Based on the security of the RCLSS scheme, the proposed CRA-aided authentication scheme with period-limited privileges is provable secure under active attacks.*

Proof. Assume that an adversary E can break the proposed scheme. We will use E to construct an algorithm F that wins each of the UF-RCLSS-ACMA games (Games I, II and III) of the RCLSS scheme, in which F plays both roles of the adversary in a UF-RCLSS-ACMA game and a challenger in the CRA-aided authentication scheme. Now, F sets $C_{pub,j} = P_{pub}$, $T_1 = H_1(TS, ID, PK_{ID}, P_{pub}, C_{pub,j}, i) = H_1(TS, ID, PK_{ID}, P_{pub}, i)$ and $T_2 = H_2(TS, ID, PK_{ID}, P_{pub}, C_{pub,j}, i) = H_2(TS, ID, PK_{ID}, P_{pub}, i)$, where H_1 and H_2 are random oracles controlled by the challenger. It is obvious that if the adversary E can forge a valid authentication request $(TS, ID, PK_{ID}, C_{pub,j}, i, \sigma)$, F may forge a signature $(m = TS, ID, PK_{ID}, i, \sigma)$ to win the UF-RCLSS-ACMA games (Games I, II and III) of the RCLSS scheme. This violates Theorems 1, 2 and 3. □

8. Conclusions

In the article, we proposed the first provably secure RCLSS scheme in the random oracle model under the computational Diffie–Hellman assumption. Comparisons with previously proposed schemes were made to demonstrate the advantages of our scheme in terms of signature size, the security and the revocation mechanism. In the security model, an adversary can obtain signature in the *Super sign* query phase without providing the corresponding secret value. (The challenger does not hold the corresponding secret value of the user, either). This is the strongest one among all the adversaries considered in CLS or RCLS schemes. Furthermore, our scheme provides the shortest signature size and strongest security level. So, the proposed RCLSS scheme is well suited for low-bandwidth communication environments with high-level security. Finally, based on the proposed RCLSS scheme, we constructed a CRA-aided authentication scheme with period-limited privileges for mobile multi-server environment.

Acknowledgements. The authors would like to thank anonymous referees for their invaluable comments and constructive suggestions. This research was partially supported by Ministry of Science and Technology, Taiwan, R.O.C., under contract no. MOST103-2221-E-018-022-MY2.

References

Al-Riyami, S.S., Paterson, K.G. (2003). Certificateless public key cryptography. In: *Proceedings of Asiacrypt'03*, *LNCS*, Vol. 2894, pp. 452–473.

Bellare, M., Rogaway, P. (1993). Random oracles are practical: a paradigm for designing efficient protocols. In: *Proceedings of CCS'93*. ACM, pp. 62–73.

Chen, Y.C., Tso, R., Susilo, W., Huang, X., Horng, G. (2013). Certificateless signatures: structural extensions of security models and new provably secure schemes. *Cryptology ePrint Archive, Report 2013/193*. Available at: http://eprint.iacr.org/2013/193.pdf.

Cheng, L., Wen, Q., Jin, Z.P., Zhang, H. (2013). On the security of a certificateless signature scheme in the standard model. *Cryptology ePrint Archive, Report, 2013/153, 2013*. http://eprint.iacr.org/2013/153.

Choi, K., Park, J., Lee, D. (2011). A new provably secure certificateless short signature scheme. *Computers and Mathematics with Applications*, 61(7), 1760–1768.

Chuang, Y.H., Tseng, Y.M. (2012). Towards generalized ID-based user authentication for mobile multi-server environment. *International Journal of Communication Systems*, 25(4), 447–460.

Dent, A.W. (2008). A survey of certificateless encryption schemes and security models. *International Journal of Information Security*, 7(5), 349–377.

Du, H., Wen, Q. (2009). Efficient and provably-secure certificateless short signature scheme from bilinear pairings. *Computer Standards and Interfaces*, 31(2), 390–394.

Du, H., Wen, Q. (2014). Security analysis of two certificateless short signature schemes. *Journal of Information Security, IET*, 8(4), 230–233.

Freier, A., Karlton, P., Kocher, P. (2011). The secure sockets layer (SSL) protocol version 3.0. *IETF*, *RFC* 6101.

Housley, R., Polk, W., Ford, W., Solo, D. (2002). Internet X. 509 public key infrastructure certificate and certificate revocation list (CRL) profile. *IETF*, *RFC* 3280.

Hu, B., Wong, D., Zhang, Z., Deng, X. (2006). Key replacement attack against a generic construction of certificateless signature. In: *Proceedings of ACISP'06*, *LNCS*, Vol. 4058, pp. 235–346.

Huang, X., Susilo, W., Mu, Y., Zhang, F. (2005). On the security of a certificateless signature scheme from asiacrypt 2003. In: *Proceedings of CANS 2005*, *LNCS*, Vol. 3810, pp. 13–25.

Huang, X., Mu, Y., Susilo, W., Wong, D., Wu, W. (2007). Certificateless signature revisited. In: *Proceedings of ACISP'06*, *LNCS*, Vol. 4586, pp. 308–322.

Hung, Y.H., Huang, S.S., Tseng, Y.M., Tsai, T.T. (2015). Certificateless signature with strong unforgeability in the standard model. *Informatica*, 26(4), 663–684.

Kaufman, C., Hoffman, P., Nir, Y., Eronen, P., Kivinen, T. (2014). Internet key exchange protocol version 2 (IKEv2). *IETF*, *RFC* 7296.

Libert, B., Quisquater, J.J. (2006). On constructing certificateless cryptosystems from identity based encryption. In: *Proceedings of PKC*, *LNCS*, Vol. 3958, pp. 474–490.

Scott, M., Costigan, N., Abdulwahab, W. (2006). Implementing cryptographic pairings on smartcards. In: *Proceedings of Cryptographic Hardware and Embedded Systems – CHES 2006*, *LNCS*, Vol. 4249, pp. 134–147.

Shamir, A. (1984). Identity-based cryptosystems and signature schemes. In: *Proceedings of Crypto'84*, *LNCS*, Vol. 196, pp. 47–53.

Shen, L., Zhang, F., Sun, Y. (2013). Efficient revocable certificateless encryption secure in the standard model. *The Computer Journal*, 57(4), 592–601. doi:10.1093/comjnl/bxt040.

Shim, K. (2009). Breaking the short certificateless signature scheme. *Information Sciences*, 179(3), 303–306.

Sun, Y., Zhang, F., Shen, L. (2014). A revocable certificateless signature scheme. *Journal of Computer*, 9(8), 1843–1850.

Tsai, T.T., Tseng, Y.M. (2015). Revocable certificateless public key encryption. *IEEE Systems Journal*, 9(3), 824–833.

Tsai, T.T., Huang, S.S., Tseng, Y.M. (2014a). Secure certificateless signature with revocation in the standard model. *Mathematical Problems in Engineering*, 2014. Article ID 728591.

Tsai, T.T., Tseng, Y.M., Wu, T.Y. (2014b). "RHIBE: constructing revocable hierarchical ID-based encryption from HIBE". *Informatica*, 25(2), 299–326.

Tseng, Y.M., Tsai, T.T. (2012). Efficient revocable ID-based encryption with a public channel. *The Computer Journal*, 55(4), 475–486.

Tseng, Y.M., Wu, T.Y., Wu, J.D. (2008). A pairing-based user authentication scheme for wireless clients with smart cards. *Informatica*, 19(2), 285–302.

Tso, R., Huang, X., Susilo, W. (2012). Strongly secure certificateless short signatures. *The Journal of Systems and Software*, 85(6), 1409–1417.

Xiong, H., Qin, Z., Li, F. (2008). An improved certificateless signature scheme secure in the standard model. *Fundamenta Informaticae*, 88(1–2), 193–206.

Yang, G., Tan, C.H. (2011). Certificateless public key encryption: a new generic construction and two pairing-free schemes. *Theoretical Computer Science*, 412(8–10), 662–674.

Yu, Y., Mu, Y., Wang, G., Xia, Q., Yang, B. (2012). Improved certificateless signature scheme provably secure in the standard model. *IET Information Security*, 6(2), 102–110.

Zhang, L., Zhang, F. (2008). A new provably secure certificateless signature scheme. In: *IEEE ICC'08*, pp. 1685–1689.

Y.-H. Hung received the BS degree from the Department of Mathematics, National Changhua University of Education, Taiwan, in 1999. He received the M.S. degree from the Department of Applied Mathematics, National Hsinchu University of Education, Taiwan, in 2008. He is currently a Ph.D. candidate in the Department of Mathematics, National Changhua University of Education, Taiwan. His research interests include applied cryptography and pairing-based cryptography.

Y.-M. Tseng is currently a Professor in the Department of Mathematics, National Changhua University of Education, Taiwan. He is a member of IEEE Computer Society, IEEE Communications Society and the Chinese Cryptology and Information Security Association (CCISA). In 2006, his paper received the Wilkes Award from The British Computer Society. He has published over one hundred scientific journal and conference papers on various research areas of cryptography, security and computer network. His research interests include cryptography, network security, computer network and mobile communications. He serves as an editor of several international journals.

S.-S. Huang is currently a Professor in the Department of Mathematics, National Changhua University of Education, Taiwan. His research interests include number theory, cryptography, and network security. He received his Ph.D. from the University of Illinois at Urbana-Champaign in 1997 under the supervision of Professor Bruce C. Berndt.

Atšaukiamoji besertifikatė trumpo parašo schema ir jos naudojimas autentifikacijai

Ying-Hao HUNG, Yuh-Min TSENG, Sen-Shan HUANG

Besertifikatė trumpo parašo schema (BTPS) turi trumpo parašo ir besertifikuotumo privalumus. Ji panaikina sertifikato būtinybę ir išsprendžia raktų deponavimo problemą identifikacija paremtų parašų atveju. BTPS sumažina komunikacijos plotį. Iki šiol nėra darbų, tiriančių atšaukimo problemą besertifikatėse trumpo parašo schemose. Šiame darbe siūlome pirmą BTPS, turinčią atšaukimo funkciją. Remdamiesi skaičiuojamąja Diffie–Hellman prielaida, rodome, kad mūsų schema turi stiprų nesuklastojamumą prieš adaptyviai parinktas pranešimo atakas. Mūsų schema turi trumpiausią parašo ilgį ir yra efektyviai realizuojama. Ji tinka atšaukimo tarnybai, administruojančiai debesų kompiuteriją.