A revocable certificateless signature scheme

Yinxia Sun[1,2], Futai Zhang[1,2], Limin Shen[1,2], and Robert H. Deng[3]

[1]School of Computer Science and Technology, Nanjing Normal University, Nanjing 210023, China
[2]Jiangsu Engineering Research Center on Information Security and Privacy Protection Technology, Nanjing 210023, China
[3]School of Information Systems, Singapore Management University, 80 Stamford Road, Singapore 178902

Abstract. Certificateless public key cryptography (CLPKC), with properties of no key escrow and no certificate, has received a lot of attention since its invention. However, membership revocation in certificateless cryptosystem still remains a non-trivial problem: the existing solutions are not practical for use due to either a costly mediator or enormous computation (secret channel). In this paper, we present a new approach to revocation in CLPKC with a concrete construction of a revocable certificateless signature (RCLS) scheme. In our scheme, a user's private key is composed of three parts: an initial partial private key, a time key and a secret value. The transmission of updated-key requires only a public channel, which makes our RCLS scheme more efficient than other methods. We first provide formal definition and security model for a RCLS scheme. The new scheme is proved secure in the random oracle model, based on the Computational Diffie-Hellman problem.

keywords: revocation, certificateless signature, Computational Diffie-Hellman problem, random oracle model.

1 Introduction

According to the way to authenticate public keys, there are mainly tree kinds of public key cryptosystems. The traditional public key cryptosystem (TPKC) uses a certificate to bind a public key with its user's identity. However, the issues associated with certificate management are quite complicated and expensive. In 1984, Shamir proposed a new public key system named "Identity-based Cryptography"(IBC)[16]. In IBC, a user's public key is his/her unique identity, thus eliminating the need for certificate. A users' private key is generated fully by the Private Key Generator (PKG). This induces the widely known key escrow problem. To solve this problem as well as to preserve the property of "certificate free", Al-Riyami and Paterson presented "Certificateless Public Key Cryptography" (CLPKC) [2]. In CLPKC, the Key Generation Center (KGC) and a user cooperates to generate a private key; the corresponding public key does not require a certificate to guarantee its authenticity. Since then, more and more attention has been paid to CLPKC. The first certificateless signature scheme was proposed in [2]. Unfortunately, it is insecure [8]. To date, there have been a lot of research work on certificateless signature (CLS) such as [22][7][8][21][20][24][11][6][9], though many of them suffer from security weakness.

For a public key cryptosystem to be applied in practice, an efficient revocation mechanism is absolutely necessary. The reason is that some private keys may become compromised. It is no longer secure for the owners of those compromised private keys. Traditionally, this problem is resolved by using certificate revocation list (CRL), online certificate status protocol (OCSP)[15], Novomodo [14] and SEM [3]. In the identity-based system, Boneh and Franklin [4] suggested a method that the PKG generates private keys for all non-revoked users periodically. Libert and Quisquater [12] applied the SEM [3] architecture to the Boneh-Franklin identity-based encryption (IBE) to obtain instantaneous revocation. In 2008, Boldyreva et al [5] utilized a binary tree to present a revocable identity based encryption scheme, which was later improved by Libert and Vergnaud [13]. In 2012, Tseng and Tsai presented a revocation mechanism by only using a public channel for key-update, and constructed a revocable identity based encryption scheme [19] and a revocable identity based signature scheme [18].

Previously, one solution to revocation in CLPKC is to employ an on-line mediator called SEM (Security Mediator) [17][10][23]. In this kind of mechanism, a user's partial private key, generated by KGC, is divided into two pieces, one of which is delivered to the user while the other is passed to the SEM. All these communications are over confidential channels. In addition, the SEM has to keep large amount of secret keys, which introduces more opportunities for attackers to compromise. Another method is to generate users' partial private keys at regular time periods [1] [17]. When a user's private key is compromised or a user leaves a position of an organization, KGC just stops the partial-private-key-update. Yet it requires all newly produced partial private keys are transmitted over expensive secret channels (between the PKG and the users).

Our Contributions. Inspired by [19] and [18], this paper presents a new and practical approach to revocation in CLPKC with a concrete construction of a revocable certificateless signature (RCLS) scheme. In our approach, a user's private key is made up of three parts: an initial partial private key, a time key and a secret value. The time key, updated in every time period, is transmitted *over a public channel*, while the initial partial private key remains unchanged. To revoke a user, KGC just stops issuing new time keys for that user. Without a time key, the user is unable to perform decryption or signing. Featuring no secret channel for key-update and no mediator, our scheme is much more efficient than previous solutions. We first give a formal definition and security model for revocable certificateless signature schemes. Based on the Bilinear Diffie-Hellman assumption, our RCLS scheme is proved existentially unforgeable in the random oracle model.

2 Definitions

2.1 Revocable certificateless signature

In this section, we define the framework of a revocable certificateless signature (RCLS). It is slightly different from the conventional CLS definition in a sense that the partial private key is added one more part called *time key*. The time key, issued by KGC, is transmitted via a public channel. When a user misbehaves or looses the private key, KGC just stops issuing new time keys for that user. A revocable certificateless signature scheme consists of the following eight algorithms:

- Setup: With a security parameter as input, this algorithm creates a list of system parameters `params` and a master key `mk`.
- Extract-Initial-Partial-Private-Key: With `params`, `mk` and an identity ID as input, this algorithm produces a partial private key D_{ID}. D_{ID} is then transmitted to the user via a secret channel.
- Update-Time-Key: With `params`, `mk`, an identity ID and a time period t as input, this algorithm generates a time key D_t. D_t is then transmitted to the user via a public channel.
- Set-Secret-Value: With `params` and ID as input, this algorithm outputs a secret value s_{ID}.
- Set-Private-Key: With `params`, D_{ID}, D_t and s_{ID} as input, this algorithm sets a private key SK_{IDt}.
- Set-Public-Key: With `params` and s_{ID} as input, this algorithm sets a public key PK_{ID}.
- Sign: With `params`, SK_{IDt}, ID, t and a message M as input, this algorithm produces a signature σ.
- Verify: With `params`, PK_{ID}, ID, t and a message/signature pair (M, σ) as input, this algorithm verifies the signature to output "accept" or "reject".

2.2 Security Model

As we know, certificateless schemes should be secure even if adversaries hold partial secret information (secret value or partial private key) of the target private key. So, two types of adversaries are considered against a certificateless scheme. A Type I adversary can replace a user's public key with a new value of its choice; a Type II adversary has knowledge of system master secret key (but cannot replace any public key). In this paper, we first extend the two types of adversaries to the setting of revocable certificateless signature and present a new type of adversary: a revoked user. For a user to be attacked, Type I adversaries have no knowledge of the initial partial private key; Type II adversaries do not have access to the secret value and the new adversary (a revoked user) lacks time keys.

Let $\mathcal{A}_I$, $\mathcal{A}_{II}$ and $\mathcal{A}_{re}$ denote a Type I, a Type II adversary and a revoked-user adversary, respectively. We consider three games Game I, Game II and Game III where $\mathcal{A}_I$, $\mathcal{A}_{II}$ and $\mathcal{A}_{re}$ interact with their challengers. Note that the challengers will keep a history of query-answer in these games.

Game I (for a Type I adversary)

- **Setup**: The challenger runs the Setup algorithm to generate a master secret key `mk` as well as a list of system public parameters `params`. `params` is given to the adversary $\mathcal{A}_I$ and `mk` is kept secret.

  $\mathcal{A}_I$ can make various queries described as follows.
- **Queries**:
  Initial Partial Private Key Extraction query $\mathsf{IPPK}(ID)$: The challenger runs Extract-Initial-Partial-Private-Key and obtains the initial partial private key D_{ID} which is then returned to $\mathcal{A}_I$.
  Time Key query $\mathsf{TK}(ID, t)$: The challenger runs Update-time-key to generate the time key D_t, then returns it to $\mathcal{A}_I$.
  Secret Value query $\mathsf{SV}(ID)$: The challenger runs Set-Secret-Value to get s_{ID}, then returns it to $\mathcal{A}_I$.
  Public Key request $\mathsf{PK}(ID)$: The challenger runs Set-Public-Key to get the public key PK_{ID} which is then delivered to $\mathcal{A}_I$.
  Public Key Replacement: The adversary $\mathcal{A}_I$ can replace any public key with a value of its own choice. The current public key is used in any subsequent computation or response according to the adversary's queries.
  Signature query $\mathsf{Sign}(M, ID, t)$: The challenger responds with a signature of M by using the correct private key of ID in the time period t.
- **Forge**: At the end of the game, $\mathcal{A}_I$ outputs a message/signature pair on behalf of a target identity ID^* in some time period t^*.

Game II (for a Type II adversary)

- **Setup**: The challenger runs the Setup algorithm to generate a list of system parameters `params` and a master key `mk`. Both `params` and `mk` are given to the adversary $\mathcal{A}_{II}$.

  Since $\mathcal{A}_{II}$ knows `mk`, it can compute any initial partial private key and any time key. $\mathcal{A}_{II}$'s various queries and the challenger's responses to them are as follows:
- **Queries**:
  Secret Value query $\mathsf{SV}(ID)$: The challenger runs Set-Secret-Value to get s_{ID} which is then returned to $\mathcal{A}_I$.
  Public Key request $\mathsf{PK}(ID)$: The challenger runs Set-Public-Key to obtain the public key PK_{ID} which is then sent to $\mathcal{A}_{II}$.
  Signature query: $\mathsf{Sign}(M, ID, t)$: The challenger responds with a signature of M for the user ID in the time period t.
- **Forge**: At the end of the game, $\mathcal{A}_{II}$ outputs a message/signature pair on behalf of a target identity ID^* in some time period t^*.

Game III (for a revoked user)

– **Setup**: The challenger runs Setup to produce a list of system parameters `params` and a master secret key `mk`. It gives `params` to the adversary $\mathcal{A}_{re}$.

  $\mathcal{A}_{re}$ can make various queries and the challenger responds to them.
– **Queries**:
  Initial Partial Private Key Extraction query IPPK(ID): The Challenger runs the algorithm Extract-Initial-Partial-Private-Key to get the initial partial private key D_{ID}, then returns it to $\mathcal{A}_{re}$.
  Time Key query TK(ID, t): The Challenger runs the algorithm Update-time-key to obtain the time key D_t, then returns it to $\mathcal{A}_{re}$.
  Secret Value query SV(ID): The challenger runs the algorithm Set-Secret-Value to produce a secret value s_{ID} as the answer to this query.
  Public Key request PK(ID): The challenger runs the algorithm Set-Public-Key to generate the public key PK_{ID} as the answer.
  Signature query Sign(M, ID, t): The challenger responds with a signature of M under the identity ID and the time period t.
– **Forge**: At the end of the game, $\mathcal{A}_I$ outputs a message/signature pair on behalf of a target entity ID^* in a time period t^*.

The adversary $\mathcal{A}_i$'s advantage in the above games is defined by the probability that $\mathcal{A}_i$ wins, $i \in \{I, II, re\}$. A RCLS scheme is said to be existentially unforgeable against chosen message attacks (EUF-CMA secure) if no probabilistic polynomial-time adversary has non-negligible advantage in the above games.

2.3 Difficult Problem

In this section, we review the definitions of bilinear pairing and Computational Diffie-Hellman problem.

Bilinear Pairing. Let $\mathbb{G}_1$ be an additive cyclic group with P a generator. Let $\mathbb{G}_2$ be a multiplicative cyclic group. Both groups are of prime order p. A *bilinear pairing* is a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ satisfying the following properties:

1. Bilinearity: given $Q, W, Z \in \mathbb{G}_1$, we have $e(Q, W + Z) = e(Q, W) \cdot e(Q, Z)$ and $e(Q + W, Z) = e(Q, Z) \cdot e(W, Z)$.
2. Non-degeneracy: $e(P, P) \neq 1_{\mathbb{G}_2}$.
3. Computability: for any $Q, W \in \mathbb{G}_1$, $e(Q, W)$ can be computed efficiently.

The computational problem below is defined in the bilinear group $(\mathbb{G}_1, \mathbb{G}_2, p, P, e)$.

Computational Diffie-Hellman (CDH) problem. *Given* (aP, bP)*, where* a, b *are uniformly chosen from* $\mathbb{Z}_q^*$*, compute* abP.

3 The construction

The concrete construction of our revocable certificateless signature scheme is as follows.

- Setup: Let G_1 be an additive cyclic group P of prime order p. P is a generator of G_1. Let G_2 be a multiplicative cyclic group of prime order p. $e : G_1 \times G_1 \longrightarrow G_2$ is a bilinear pairing. Choose a random $s \in Z_p^*$ and compute $P_0 = sP$. There are four hash functions: $H_1 : \{0,1\}^* \to G_1$, $H_2 : \{0,1\}^* \to G_1$, $H_3 : \{0,1\}^* \to G_1$, $H_4 : \{0,1\}^* \to G_1$. The system public parameters are $(p, G_1, G_2, P, e, P_0, H_1, H_2, H_3, H_4)$. The master secret key is s.
- Extract-Initial-Partial-Private-Key: Inputting an identity ID, this algorithm computes $Q_{ID} = H_1(ID)$ and the partial private key $D_{ID} = sQ_{ID}$, then transmits D_{ID} to the user via a secret channel.
- Update-Time-Key: Inputting an identity ID and a time period t, this algorithm computes $Q_t = H_1(ID, t)$ and the time key $D_t = sQ_t$, then transmits D_t to the user via a public channel.
- Set-Secret-Value: This algorithm produces a secret value x_{ID} which is randomly chosen from Z_p^*.
- Set-Private-Key: For a user with identity ID at the time period t, the full private key SK_{IDt} is expressed as (D_{IDt}, x_{ID}), where $D_{IDt} = D_{ID} + D_t$.
- Set-Public-Key: The public key of the user is $PK_{ID} = x_{ID}P$.
- Sign: This algorithm takes as input a message M, a time t and a signer's private key SK_{IDt}, then does the following:
  1. Choose $r \in Z_p^*$ at random and compute $U = rP$.
  2. Compute $V = D_{IDt} + rH_3(M, ID, t, PK_{ID}, U) + x_{ID}H_4(M, ID, t, PK_{ID})$.
  3. Output the signature $\sigma = (U, V)$.
- Verify: This algorithm takes as input a message/signature pair $(M, \sigma = (U, V))$, a time period t and the signer's public key ID and PK_{ID}, then checks whether the equation

  $$e(V, P) =$$

  $$e(Q_{ID} + Q_t, P_0)e(H_3(M, ID, t, PK_{ID}, U), U)e(H_4(M, ID, t, PK_{ID}), PK_{ID})$$

  holds. If yes, output "accept"; otherwise, output "reject".

4 Security and Efficiency Analysis

Our RCLS scheme is existentially unforgeable against chosen message attacks from all adversaries. We prove the security by the following three theorems.

Theorem 1 Suppose H_1, H_2, H_3, H_4 are random oracles and there exists a Type I EUF-CMA adversary $\mathcal{A}_I$ against the RCLS scheme with advantage ϵ when running in time t, making q_{ippk} initial partial private key queries, q_{tk} time key queries, q_{pk} public key queries, q_{sign} signature queries, and q_i random oracle queries to H_i $(1 \leqslant i \leqslant 4)$. Then, there exists an algorithm $\mathcal{B}$ to solve the CDH problem with advantage $\epsilon' \geqslant \frac{1}{q_2}\epsilon$ and running in time $t' = t + (q_1 + q_2 + q_3 + q_4 + q_{ippk} + q_{tk} + q_{pk} + 3q_{sign})(T_S + O(1))$, where T_S denotes the time for computing scalar multiplication.

Proof. Let (P, aP, bP) be a random instance of the CDH problem. Next we show how to construct an algorithm B to solve the CDH problem by interacting with the adversary A_I.

At the beginning, B provides A_I with the system parameters $(p, G_1, G_2, P, e, P_0 = aP, H_1, H_2, H_3, H_4)$ described as in the concrete scheme. Here, we view the hash functions $H_i, (i = 1, 2, 3, 4)$ as random oracles controlled by B. B chooses an index $z \in [1, q_2] \cap \mathbb{Z}$ uniformly at random. Suppose the zth query is on (ID^*, t^*).

A_I may make oracle queries on hashes, initial partial private keys, time keys, secret values, public keys and signatures. Also, A_I can replace public keys.

H_1 queries: B maintains an H_1 list of tuples (ID_i, Q_i, h_{1i}). On receiving an H_1 query on ID_i, B performs the following steps:

- if $ID_i = ID^*$, set $Q_i = bP - H_2(ID^*, t^*)$;
- else, B chooses $h_{1i} \in Z_p^*$ at random, computes $Q_i = H_1(ID_i) = h_{1i}P$;
- Add the corresponding tuple to the list.

H_2 queries: B maintains an H_2 list of tuples $(ID_i, t_j, Q_{ij}, h_{2ij}, z)$. z denotes the number of this query among all H_2 queries. On receiving an H_2 query on (ID_i, t_j), B selects $h_{2ij} \in Z_p^*$ at random, computes $Q_{ij} = H_1(ID_i, t_j) = h_{2ij}P$;

H_3 queries: B maintains an H_3 list of tuples $(M, ID_i, t_j, PK_{IDi}, U, h_{3ij})$. On receiving an H_3 query on $(M, ID_i, t_j, PK_{IDi}, U)$, B chooses $h_{3ij} \in Z_p^*$ at random, computes $H_3(M, ID_i, t_j, PK_{IDi}, U) = h_{3ij}P$, and add the corresponding tuple to the list.

H_4 queries: B maintains an H_4 list of tuples $(M, ID_i, t_j, PK_{IDi}, h_{4ij})$. On receiving an H_4 query on (M, ID_i, t_j, PK_{IDi}), B chooses $h_{4ij} \in Z_p^*$ at random, computes $H_4(M, ID_i, t_j, PK_{IDi}) = h_{4ij}P$, and add the corresponding tuple to the list.

From now on, we assume that A_I always makes the appropriate H_1 and H_2 queries before making other related queries as described below.

Initial Partial Private Key Extraction queries: B maintains an initial partial private key list of tuples (ID_i, D_i). On receiving such a query on an identity ID_i,

- if $ID_i = ID^*$, B aborts the game;
- else, B calculates $D_i = aH_1(ID_i) = h_{1i}aP$ as the initial partial private key.

Time Key queries: B maintains a time key list of tuples (ID_i, t_j, D_{ij}). On receiving such a query on an identity-time pair (ID_i, t_j), B computes $D_{ij} = aH_1(ID_i, t_j) = h_{2ij}aP$ as the time key. Send D_{ij} to A_{in} and add the tuple (ID_i, t_j, D_{ij}) to the list.

Secret Value queries: Any secret value of any identity can be queried by the adversary. B just responds with an x which is randomly chosen from Z_p^*.

Public Key queries: When receiving a public key query, B responds with $PK_{ID} = xP$ where x is the secret value.

Public Key Replacement: A_I can replace any public key with a new value chosen by itself.

Signature queries: When receiving a signature query on (M, ID, t),

– if $ID \neq ID^*$ and the public key of ID remains unchanged, B runs the Sign algorithm normally to produce a signature.
– if $ID = ID^*$ or the public key of ID has been replaced, B yields a signature in the following way:
  - Pick $u, v \in Z_p^*$ at random.
  - Compute $U = uPK_0$ and $V = vPK_0 + h_4PK_{ID}$.
  - The signature is $\sigma = (U, V)$. Here, we set $H_3(M, ID, t, PK_{ID}, U) = u^{-1}(vP - H_1(ID) - H_2(ID, t))$. Note that if there has been an tuple with the form $(M, ID, t, PK_{ID}, U, ?)$, we choose another $u \in Z_p^*$ and repeat this signature procedure.

Forge: Finally, A_I outputs a signature $\sigma^* = (U^*, V^*)$ of ID^* on a message M^* at the time period t^*. If σ^* is valid, it should pass the verification:

$$e(V^*, P) = e(Q_{ID^*} + Q_{t^*}, P_0)e(H_3(), U^*)e(H_4(), PK_{ID^*}),$$

where $H_3()$ is short for $H_3(M^*, ID^*, t^*, PK_{ID^*}, U^*)$ and $H_4()$ is short for $H_4(M^*, ID^*, t^*, PK_{ID^*})$. Search the H_3 and H_4 list for $H_3(M^*, ID^*, t^*, PK_{ID^*}, U^*) = h_3P$ and $H_4(M^*, ID^*, t^*, PK_{ID^*}) = h_4P$ respectively. Obviously, the above equation can be transformed into

$$e(V^* - h_3U^* - h_4PK_{ID^*}, P) = e(abP, P).$$

Now, it is easy for B to obtain the CDH solution $abp = V^* - h_3U^* - h_4PK_{ID^*}$.

Analysis. It is not difficult for us to obtain the advantage for $\mathcal{B}$ to solve the CDH problem $\epsilon' \geqslant \frac{1}{q_2}\epsilon$.

The running time of $\mathcal{B}$ is bounded by $t' = t + (q_1 + q_2 + q_3 + q_4 + q_{ippk} + q_{tk} + q_{pk} + 3q_{sign})(T_S + O(1))$, where T_S denotes the time for doing scalar multiplication.

Theorem 2 Suppose H_1, H_2, H_3, H_4 are random oracles and there exists a Type II EUF-CMA adversary $\mathcal{A}_{II}$ against the RCLS scheme with advantage ϵ when running in time t, making q_{pk} public key queries, q_{sign} signature queries, and q_i random oracle queries to H_i $(1 \leqslant i \leqslant 4)$. Then, there exists an algorithm $\mathcal{B}$ to solve the CDH problem with advantage $\epsilon' \geqslant \frac{1}{q_1}\epsilon$ and running in time $t' = t + (q_1 + q_2 + q_3 + q_4 + q_{pk} + 3q_{sign})(T_S + O(1))$, where T_S denotes the time for computing scalar multiplication.

Proof. Let (P, aP, bP) be a random instance of the CDH problem. Next we show how to construct an algorithm B to solve the CDH problem by interacting with the inside adversary A_{II}.

At the beginning, B chooses a random $s \in Z_p^*$ as the master secret key and provides A_{II} with s and the system parameters $(p, G_1, G_2, P, e, P_0 = sP, H_1, H_2, H_3, H_4)$ described as in the concrete scheme. Here, we view the hash functions $H_i, (i = 1, 2, 3, 4)$ as random oracles controlled by B. B chooses an index I uniformly at random from $[1, q_1] \cap \mathbb{Z}$.

A_{II} can make queries to $H_i (1 \leqslant i \leqslant 4)$ oracles.

H_1 queries: B maintains an H_1 list of tuples (ID_i, Q_i, h_{1i}). On receiving an H_1 query on ID_i, B chooses $h_{1i} \in Z_p^*$ at random, computes $Q_i = H_1(ID_i) = h_{1i}P$, and add the corresponding tuple to the list.

H_2 queries: B maintains an H_2 list of tuples $(ID_i, t_j, Q_{ij}, h_{2ij})$. On receiving an H_2 query on (ID_i, t_j), B chooses $h_{2ij} \in Z_p^*$ at random, computes $Q_{ij} = H_2(ID_i, t_j) = h_{2ij}P$, and add the corresponding tuple to the list.

H_3 queries: B maintains an H_3 list of tuples $(M, ID_i, t_j, PK_{IDi}, U, h_{3ij})$. On receiving an H_3 query on $(M, ID_i, t_j, PK_{IDi}, U)$, B chooses $h_{3ij} \in Z_p^*$ at random, computes $H_3(M, ID_i, t_j, PK_{IDi}, U) = h_{3ij}P$, and add the corresponding tuple to the list.

H_4 queries: B maintains an H_4 list of tuples $(M, ID_i, t_j, PK_{IDi}, h_{4ij})$. On receiving an H_4 query on (M, ID_i, t_j, PK_{IDi}), B chooses $h_{4ij} \in Z_p^*$ at random, computes $H_4(M, ID_i, t_j, PK_{IDi}) = h_{4ij}bP$, and add the corresponding tuple to the list.

Since A_{II} knows the master secret key, it can compute all initial partial private keys and all time keys. It can request secret values, public keys and signatures. Assume that A_{II} always makes the appropriate H_1 and H_2 queries before making other related queries as described below.

Secret Value queries: B maintains a secret value list of tuples (ID_i, x_i). On receiving such a query on an identity ID_i, B searches the list: if there has been a corresponding tuple, return the secret value; otherwise, do the following:

- if $i = I$, abort the game.
- if $i \neq I$, randomly choose $x_i \in Z_p^*$ as the secret value, and add (ID_i, x_i) to the list.

Public Key queries: B maintains a secret value list of tuples (ID_i, PK_i). On receiving such a query of an identity ID_i, B searches the list: if there has been a corresponding tuple, return the public key; otherwise, do the following:

- if $i = I$, return $PK_I = aP$.
- if $i \neq I$, B searches the secret value list for an x_i and computes $PK_i = x_iP$. If there is not a matched secret value with ID_i, B chooses $x_i \in Z_p^*$ and computes $PK_i = x_iP$.Add (ID_i, x_i) to the secret value list and (ID_i, PK_i) to the public key list.

Signature queries: On receiving a signature query of (M, ID, t), B acts as follows:

- if $ID \neq ID^*$, run the sign algorithm normally.
- else, B selects $u, v \in Z_p^*$ at random, computes $U = uPK_{ID}$ and $V = vPK_{ID} + D_{IDt}$. The signature is $\sigma = (U, V)$. Here, we set $H_3(M, ID, t, PK_{ID}, U) = u^{-1}(vP - H_4(M, ID, t, PK_{ID})$. Note that if there has been an tuple with the form $(M, ID, t, PK_{ID}, U, ?)$, we choose another $u \in Z_p^*$.

Forge: Finally, A_{II} outputs a signature $\sigma^* = (U^*, V^*)$ of ID^* on a message M^* at a time period t^*. If σ^* is valid, it should pass the verification:

$$e(V^*, P) = e(Q_{ID^*} + Q_{t^*}, P_0)e(H_3(), U^*)e(H_4(), PK_{ID^*}),$$

where $H_3(M^*, ID^*, t^*, PK_{ID^*}, U^*)$ is short for $H_3()$ and $H_4(M^*, ID^*, t^*, PK_{ID^*})$ is short for $H_4()$. Search the H_3 and H_4 list for $H_3(M^*, ID^*, t^*, PK_{ID^*}, U^*) = h_3P$ and $H_4(M^*, ID^*, t^*, PK_{ID^*}) = h_4bP$ respectively. Obviously, the above equation can be transformed into

$$e(V^* - D_{ID^*t^*} - h_3U^*, P) = e(h_4abP, P).$$

Now, it is easy for B to obtain the CDH solution $abp = h_4^{-1}(V^* - D_{ID^*t^*} - h_3U^*)$.

Analysis. It is not difficult for us to obtain the advantage for $\mathcal{B}$ to solve the CDH problem $\epsilon' \geqslant \frac{1}{q_1}\epsilon$.

The running time of $\mathcal{B}$ is bounded by $t' = t + (q_1 + q_2 + q_3 + q_4 + q_{pk} + 3q_{sign})(T_S + O(1))$, where T_S denotes the time for doing scalar multiplication.

Theorem 3 Suppose H_1, H_2, H_3, H_4 are random oracles and there exists a revoked user $\mathcal{A}_{re}$ who can break the EUF-CMA security of the RCLS scheme with advantage ϵ when running in time t, making q_{ippk} initial partial private key queries, q_{tk} time key queries, q_{pk} public key queries, q_{sign} signature queries, and q_i random oracle queries to H_i ($1 \leqslant i \leqslant 4$). Then, there exists an algorithm $\mathcal{B}$ to solve the CDH problem with advantage $\epsilon' \geqslant \frac{1}{q_2}\epsilon$ and running in time $t' = t + (q_1 + q_2 + q_3 + q_4 + q_{ippk} + q_{tk} + q_{pk} + 3q_{sign})(T_S + O(1))$, where T_S denotes the time for computing scalar multiplication.

Proof. Let (P, aP, bP) be a random instance of the CDH problem. Next we show how to construct an algorithm B to solve the CDH problem by interacting with the inside adversary A_{in}.

At the beginning, B provides A_{in} with the system parameters $(p, G_1, G_2, P, e, P_0 = aP, H_1, H_2, H_3, H_4)$ described as in the concrete scheme. Here, we view the hash functions $H_i, (i = 1, 2, 3, 4)$ as random oracles controlled by B. B chooses an index $z \in [1, q_2] \cap \mathbb{Z}$ uniformly at random. Suppose the zth query is on (ID^*, t^*).

A_{in} may make oracle queries on hashes, initial partial private keys, time keys, secret values, public keys and signatures.

H_1 queries: B maintains an H_1 list of tuples (ID_i, Q_i, h_{1i}). On receiving an H_1 query on ID_i, B chooses $h_{1i} \in Z_p^*$ at random, computes $Q_i = H_1(ID_i) = h_{1i}P$, and add the corresponding tuple to the list.

H_2 queries: B maintains an H_2 list of tuples $(ID_i, t_j, Q_{ij}, h_{2ij}, f)$. f denotes the number of this query among all H_2 queries. On receiving an H_2 query on (ID_i, t_j),

- if $f = z$, set $Q_{ij} = bP - H_1(ID^*)$;
- else, B chooses $h_{2ij} \in Z_p^*$ at random, computes $Q_{ij} = H_2(ID_i, t_j) = h_{2ij}P$.
- Add the corresponding tuple to the list.

H_3 queries: B maintains an H_3 list of tuples $(M, ID_i, t_j, PK_{IDi}, U, h_{3ij})$. On receiving an H_3 query on $(M, ID_i, t_j, PK_{IDi}, U)$, B chooses $h_{3ij} \in Z_p^*$ at random, computes $H_3(M, ID_i, t_j, PK_{IDi}, U) = h_{3ij}P$, and add the corresponding tuple to the list.

H_4 queries: B maintains an H_4 list of tuples $(M, ID_i, t_j, PK_{IDi}, h_{4ij})$. On receiving an H_4 query on (M, ID_i, t_j, PK_{IDi}), B chooses $h_{4ij} \in Z_p^*$ at random, computes $H_4(M, ID_i, t_j, PK_{IDi}) = h_{4ij}P$, and add the corresponding tuple to the list.

From now on, we assume that A_{in} always makes the appropriate H_1 and H_2 queries before making other related queries as described below.

Initial Partial Private Key Extraction queries: B maintains an initial partial private key list of tuples (ID_i, D_i). On receiving such a query on an identity ID_i, B calculates the initial partial private key $D_i = aH_1(ID_i) = h_{1i}aP$. Send D_i to A_{in} and add the tuple (ID_i, D_i) to the list.

Time Key queries: B maintains a time key list of tuples (ID_i, t_j, D_{ij}). On receiving such a query on an identity (ID_i, t_j), B calculates the time key $D_{ij} = aH_1(ID_i, t_j) = h_{2ij}aP$. Send D_{ij} to A_{in} and add the tuple (ID_i, t_j, D_{ij}) to the list. Note that the time key query on (ID^*, t^*) is not allowed, since it is to be challenged.

Secret Value queries: Any secret value of any identity can be queried by the adversary. B just responds with an x which is randomly chosen from Z_p^*.

Public Key queries: When receiving a public key query, B responds with $PK_{ID} = xP$ where x is the secret value.

Signature queries: When receiving a signature query on (M, ID, t), B runs the sign algorithm normally to produce a signature. Note that, the adversary cannot ask for a signature of (ID^*, t^*), since A_{re} has been revoked in this time period.

Forge: Finally, A_{II} outputs a signature $\sigma^* = (U^*, V^*)$ of ID^* on a message M^* at the time period t^*. Note that the time key for (ID^*, t^*) is never been requested. If σ^* is valid, it should pass the verification:

$$e(V^*, P) = e(Q_{ID^*} + Q_{t^*}, P_0)e(H_3(), U^*)e(H_4(), PK_{ID^*}),$$

where $H_3(M^*, ID^*, t^*, PK_{ID^*}, U^*)$ is short for $H_3()$ and $H_4(M^*, ID^*, t^*, PK_{ID^*})$ is short for $H_4()$. Search the H_3 and H_4 list for $H_3(M^*, ID^*, t^*, PK_{ID^*}, U^*) = h_3P$ and $H_4(M^*, ID^*, t^*, PK_{ID^*}) = h_4P$ respectively. Obviously, the above e-quation can be transformed into

$$e(V^* - h_3U^* - h_4PK_{ID^*}, P) = e(abP, P).$$

Now, it is easy for B to obtain the CDH solution $abp = V^* - h_3U^* - h_4PK_{ID^*}$.

Analysis. It is not difficult for us to obtain the advantage for $\mathcal{B}$ to solve the CDH problem $\epsilon' \geqslant \frac{1}{q_2}\epsilon$.

The running time of $\mathcal{B}$ is bounded by $t' = t + (q_1 + q_2 + q_3 + q_4 + q_{ippk} + q_{tk} + q_{pk} + 3q_{sign})(T_S + O(1))$, where T_S denotes the time for doing scalar multiplication.

4.1 Efficiency

As is seen, in our RCLS scheme, a user's key is made up of an initial partial private key, a time key and a secret value. Revocation is obtained by updating the time key. Different from existing solutions, the time key is transmitted over public channels. This property makes our new scheme more applicable in practice. In the table below, we make a comparison of computational cost, ciphertext-length and revocation-type of our scheme with that of a trivial revocable CLS scheme (it employs the same signing technique as ours; a user's partial private key $D_{IDt} = sH_1(ID, t)$ is generated by KGC at every time period and is transmitted via a secret channel).

Table 1. Comparison

Scheme	Sign	verify	ciphertext	revocation-type
the trivial one	3s	4p	$2\|P\|$	**secret** channel
Our Scheme	3s	4p	$2\|P\|$	**public** channel

p: pairing, s: scalar multiplication, $|P|$: the length of an element in $\mathbb{G}_1$.

In the table, "revocation-type" denotes what kind of channel is employed for updating keys. **Secret** channel indicates that both KGC and users have to do enormous computation for the secure transmission of new partial private keys. Our RCLS scheme has better performance.

5 Conclusion

How to revoke a user is a necessary problem in the application of public key cryptosystems. In this paper, we concentrate on revocation in CLPKC. On one hand, we present an efficient revocation mechanism for CLPKC. On the other hand, with our revocation mechanism we introduce a revocable certificateless signature (RCLS) scheme. In contrast to available solutions, our new construction features public channels for key-updating, avoiding the use of secret channels or a costly mediator. So, the new scheme is very efficient and is more suitable for resource-limited applications. With respect to the security of RCLS schemes, we demonstrate a reasonable security model for RCLS schemes in which the adversaries are classified into three types for the first time. The security proofs confirm that our RCLS scheme is provably secure in the random oracle model based on the CDH problem.

This work is supported by the Nature Science Foundation of China [No.61170298]; Natural Science Fund for Colleges and Universities in Jiangsu Province (No.12KJD520007); and Nanjing Normal University Foundation [2012119XGQ181].

References

1. Sattam S. Al-Riyami. Cryptographic Schemes Based on Elliptic Curve Pairings. PhD thesis, Royal Holloway, University of London, 2004.
2. S. S. Al-Riyami and K. G. Paterson. Certificateless Public Key Cryptography. Asiacrypt 2003, LNCS 2894. Berlin: Springer-Verlag, 2003. 452-473

3. D.Boneh, X.Ding, G.Tsudik, and C.Wong, A method for fast revocaiton of public key certificates and security capabilities,In Proceedings of the 10th USENIX Security Symposium, USENIX, 2001.
4. Boneh D, Franklin M. Identity-based Encryption from the Weil Pairing. In: Proc of CRYPTO 2001, LNCS Vol. 2139. Berlin: Springer-Verlag, 2001. 213-229
5. Boldyreva A, Goyal V, Kumar V. Identity-based encryption with efficient revocation. Proc. CCS 2008. ACM, 2008. 417-426
6. Peng Gong, Ping Li. Further improvement of a certificateless signature scheme without pairing. International Journal of Communication Systems. Article first published online: 22 OCT 2012. DOI: 10.1002/dac.2457
7. Gorantla, M.C., Saxena, A.: An Efficient Certificateless Signature Scheme. In: Hao, Y., Liu, J., Wang, Y.-P., Cheung, Y.-m., Yin, H., Jiao, L., Ma, J., Jiao, Y.-C. (eds.) CIS 2005, LNCS 3802. Berlin: Springer-Verlag, 2005 110C116
8. Huang, X., Susilo, W., Mu, Y., Zhang, F.: On the Security of Certificateless Signature Schemes from Asiacrypt 2003. In: Desmedt, Y.G., Wang, H., Mu, Y., Li, Y. (eds.) CANS 2005. LNCS 3810. Berlin: Springer-Verlag, 2005. 13-25
9. Xinyi Huang, Yi Mu, Willy Susilo, Duncan S. Wong, and Wei Wu. Certificateless Signatures: New Schemes and Security Models. The Computer Journal, 2012, 55(4): 457-474
10. Hak Soo Ju, Dae Youb Kim, Dong Hoon Lee, Jongin Lim, and Kilsoo Chun. Efficient Revocation of Security Capability in Certificateless Public Key Cryptography. KES 2005, LNCS 3682. Berlin: Springer-Verlag, 2005. 453C459
11. Young-Ran Lee. On the Security of Two Certificateless Signature Schemes. Advances in Intelligent and Soft Computing Volume 167, 2012. 695-702
12. B Libert, J-J Quisquater. Efficient revocation and threshold pairing based cryptosystems, Symposium on Principles of Distributed Computing-PODC 2003, 2003.
13. B Libert, D Vergnaud. Adaptive-ID secure revocable identity-based encryption. Proc. CT-RSA 2009, LNCS 5473. Berlin: Springer-Verlag, 2009. 1-15
14. Silvio Micali. Novomodo: Scalable Certificate Validation and Simplified PKI Management. In Proceedings of 1st Annual PKI Research Workshop 2002. 15-25
15. M Myers, R Ankney, A alpani, S Galperin, C Adams. X.509 Internet Public Key Infrastructure: Online Certificate Status Protocol (OCSP), RFC 2560.
16. Adi Shamir. Identity-based cryptosystems and signature schemes. In CRYPTO 1984, LNCS. Berlin: Springer-Verlag, 1984.47-53
17. Sherman SM Chow, Colin Boyd, Juan Manuel Gonzalez Nieto. Security-Mediated Certificateless Cryptography. PKC 2006, LNCS 3958, Berlin: Springer-Verlag, 2006. 508-524
18. Tung-Tso Tsai, Yuh-Min Tseng, Tsu-Yang Wu. Revocable ID-based Signature Scheme with Batch Verifications. 2012 Eighth International Conference on Intelligent Information Hiding and Multimedia Signal, 2012. 49-54
19. Y-M Tseng, T-T Tasi. Efficient revocable ID-based encryption with a public channel. The Computer Journal, 2012, 55(4): 475-486
20. Raylin Tso, Xun Yi, Xinyi Huang. Efficient and Short Certificateless Signature. The Journal of Supercomputing, 2011, 55(2): 173-191
21. W-S Yap, S-H Heng, B-M Goi. An Efficient Certificateless Signature Scheme. In: Zhou, X., Sokolsky, O., Yan, L., Jung, E.-S., Shao, Z., Mu, Y., Lee, D.C., Kim, D., Jeong, Y.-S., Xu, C.-Z. (eds.) Emerging Directions in Embedded and Ubiquitous Computing, LNCS 4097. Berlin: Springer-Verlag, 2006. 322-331
22. DH Yum, PJ Lee. Generic Construction of Certificateless Signature. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS 3108. Berlin: Springer-Verlag, 2004. 200-211

23. W-S Yap, Sherman SM Chow, S-H Heng, B-M Goi1. Security Mediated Certificateless Signatures. ACNS 2007, LNCS 4521. Berlin: Springer-Verlag, 2007. 459-477
24. Z Zhang, DS Wong, J Xu, D Feng. Certificateless Public-Key Signature: Security Model and Efficient Construction. In: Zhou, J., Yung, M., Bao, F. (eds.) ACNS 2006, LNCS 3989. Berlin: Springer-Verlag, 2006. 293-308