A Revocable Certificateless Signature Scheme

Yinxia Sun[a,b], Futai Zhang[a,b], Limin Shen[a,b]
[a] School of Computer Science and Technology, Nanjing Normal University, Nanjing 210023, China
[b] Jiangsu Engineering Research Center on Information Security and Privacy Protection Technology, Nanjing 210023, China

***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, how to revoke a user in certificateless cryptosystem still remains a 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 solve the revocation problem in CLPKC, by giving a concrete revocable certificateless signature scheme. The new scheme is more efficient than the existing solutions and is provably secure under the Computational Diffie-Hellman assumption.**

***Index Terms*— revocation, certificateless signature, existential unforgeability, Computational Diffie-Hellman problem.**

I. 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. Identity-based cryptosystem (IBC), introduced by Shamir in 1984 [14], utilizes a user's publicly known identity information as its public key. So, there is no need for a public key certificate. However, a user's private key must be fully generated by a Private Key Generater (PKG). In order to preserving the "certificate free" property of IBC without suffering from the key escrow problem, 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. Take a scenario in cloud computing as an example. Cloud computing is a distributed system where multiple cloud servers co-exist. Every cloud server has its own master secret key and public key certified by a PKI. Due to the heavy burden of certificate management, a cloud server may provide services to users via IBC playing the role of PKG. All users trust the server. As some users may want to keep their privacy from the cloud sever, they can use CLPKC by making use of the cloud sever as a partial private key generation center. A user in this cloud can utilize powerful computing resources to store sensitive data (by certificateless encryption), or to declare the authenticity of a document (by certificateless signature) shared with others.

It is widely known that a necessary issue in the practical application of a public key cryptosystem is to establish an effective revocation mechanism. Once a user's private key is compromised or the access permission is expired, the cloud center should revoke the user's current private key. Traditionally, this problem can be solved by using certificate revocation lists (CRLs), online certificate status protocol (OCSP) [12], Novomodo [11] and SEM [3]. In identity-based public key systems, Boneh and Franklin [4] suggested a method that the PKG generates private keys for all non-revoked users periodically. Libert and Quisquater [9] 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 the first scalable revocable identity based encryption scheme, which was later improved by Libert and Vergnaud [10]. In 2012, Tseng and Tsai presented an efficient revocable identity based encryption scheme [17] and a revocable identity based signature scheme [16]. In PKC 2013, Seo et al [13] made a survey of revocable identity based encryption, and presented a new realistic threat named "decryption key exposure" against revocable identity based primitives. Decryption key exposure captures the security notion that a ciphertext does not leak any information about the plaintext even if all previous decryption keys are exposed. They proposed the first scalable revocable identity based encryption scheme against decryption key exposure.

One previous solution to revocation in CLPKC is to employ an on-line mediator called SEM (Security Mediator) [15] [6] [18]. In this kind of mechanism, the KGC divides a user's partial private key 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 pieces which increases linearly with the number of users. Moreover, a user cannot do decryption/signing independently. Another one is to generate users' partial private keys at regular time periods [1] [15]. When a user needs to be revoked, KGC just stops updating its partial private key. Yet it requires all newly produced partial private keys for non-revoked users to be transmitted over expensive secret channels (between the KGC and the users). In 2013, a certificateless encryption with a

This work is supported by NSF of Jiangsu Province of China [No. BK20130908], the Nature Science Foundation of China [No.61170298], Natural Science Fund for Colleges and Universities in Jiangsu Province [No.13KJD520006] and Nanjing Normal University Foundation [2012119XGQ181].

doi:10.4304/jcp.9.8.1843-1850

revocation mechanism was presented in [8]. However, the scheme suffers from some security weakness and a low efficiency.

Our Contributions. Inspired by the revocation technique in the identity based setting, 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, we require the KGC produce for a user an initial partial private key based on the user's identity information as well as a time key corresponding to each time period. The time key is updated in every time period, and is transmitted over a public channel. To revoke a user, KGC just stops running this new algorithm for the user. Without a time key, the user is unable to correctly perform any decryption/signing. Removing the use of secret channels for key-update and without resorting to a security, our scheme offers better efficiency than previous solutions. In the rest of the paper, we first introduce the formal definition and security model for revocable certificateless signature schemes. Then, we present an efficient RCLS scheme. Based on the Computational Diffie-Hellman assumption, our RCLS scheme is proved existentially unforgeable in the random oracle model. At last, we show an approach to extend our scheme to resist the threat of decryption key exposure. (More precisely, we may call it signing key exposure.)

II. Definitions

A. Revocable certificateless signature

In this section, we define the framework for a revocable certificateless signature (RCLS) scheme. It is slightly different from the conventional definition of certificateless signature in a sense that the partial private key is divided into an initial partial private key and a *time key*. The time key is transmitted to the user via a public channel. The revocation is achieved by stopping the production of new time keys for the revoked user. A revocable certificateless signature scheme consists of the following eight algorithms:

- Setup: Taking a security parameter k as input, the KGC runs this algorithm to generate a master key `mk` and a list of public system parameters `params`.
- Extract-Initial-Partial-Private-Key: Taking `params`, `mk` and an identity ID as input, the KGC runs this algorithm to compute a partial private key D_{ID}. D_{ID} is transmitted to the user via a secret channel.
- Update-Time-Key: Taking `params`, `mk`, an identity ID and a time period t as input, the KGC runs this algorithm to produce a time key D_{IDt}. D_{IDt} is transmitted to the user via a public channel.
- Set-Secret-Value: Taking `params` and ID as input, the user with ID runs this algorithm to generate a secret value s_{ID}.
- Set-Private-Key: Taking `params`, D_{ID}, D_{IDt} and s_{ID} as input, the user runs this algorithm to set a private key SK_{IDt}.
- Set-Public-Key: Taking `params` and s_{ID} as input, the user runs this algorithm to set a public key PK_{ID}.
- Sign: Taking `params`, SK_{IDt}, ID, t and a message M as input, this algorithm outputs a signature σ.
- Verify: Taking `params`, PK_{ID}, ID, t and a signature σ as input, this algorithm verifies the signature to output "accept" or "reject".

B. Security Model

As we know, certificateless schemes should be secure even if adversaries get to know some partial secret information (secret value or partial private key) of the target identity. 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 extend the two types of adversaries to the setting of revocable certificateless signature and consider a new type of adversary: a malicious revoked user. For a target user, 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 a time key.

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 algorithm Setup to generate a master secret key `mk` and a list of public system parameters `params`. It gives `params` to the adversary $\mathcal{A}_I$ and keeps `mk` secret.
- **Queries**:
  In this phase, $\mathcal{A}_I$ may make some queries and the challenger should response with proper answers.
  Initial Partial Private Key Extraction query(ID): The challenger runs Extract-Initial-Partial-Private-Key to generate the initial partial private key D_{ID}, then returns it to $\mathcal{A}_I$.
  Time Key query(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(ID): The challenger runs Set-Secret-Value to generate s_{ID}, then returns it to $\mathcal{A}_I$.
  Public Key request(ID): The challenger runs Set-Public-Key to generate the public key PK_{ID}. It returns PK_{ID} to $\mathcal{A}_I$.
  Public Key Replacement: The adversary $\mathcal{A}_I$ can replace any public key with any value of its choice. The current public key is used by the challenger in any subsequent computation or response to $\mathcal{A}_I$'s requests.
  Signature query(M, ID, t): The challenger responds with the signature of M by using the private

key of ID in the time period t.

- **Forge**: At the end of the game, $\mathcal{A}_I$ outputs a tuple $(S*, M*, ID*, t*)$ which means that it has forged a user $ID*$'s signature $S*$ on a message $M*$ at time period $t*$. Note that $(S*, M^*, ID^*, t^*)$ should not be an output of the signature oracle.

Game II (for a Type II adversary)

- **Setup**: The challenger runs Setup to generate a master key `mk` and a list of public parameters `params`. It gives `mk` as well as `params` to the adversary $\mathcal{A}_{II}$.
- **Queries**: In this phase, the adversary may make some queries. As $\mathcal{A}_{II}$ knows `mk`, it can compute any initial partial private key and any time key.

  Secret Value query(ID): The challenger runs Set-Secret-Value to generate s_{ID}, then returns it to $\mathcal{A}_I$.
  Public Key request(ID): The challenger runs Set-Public-Key to generate the public key PK_{ID} which is then returned to $\mathcal{A}_{II}$.
  Signature query:(M, ID, t): The challenger responds with user ID's signature on message M at the time period t.
- **Forge**: At the end of the game, $\mathcal{A}_{II}$ outputs a tuple $(S*, M*, ID*, t*)$ which means that it has forged a user $ID*$'s signature $S*$ on a message $M*$ at time period $t*$. $(S*, M*, ID*, t*)$ should not be an output of the signature oracle.

Game III (for a revoked user)

- **Setup**: The challenger runs Setup to generate a master secret key `mk` and a list of public parameters `params`. It gives `params` to the adversary $\mathcal{A}_{re}$.
- **Queries**:
  In this phase, $\mathcal{A}_{re}$ may make some queries as follows:

  Initial Partial Private Key Extraction query(ID): The Challenger runs Extract-Initial-Partial-Private-Key to generate the initial partial private key D_{ID}, then returns it to $\mathcal{A}_{re}$.
  Time Key query(ID, t): The Challenger runs Update-time-key to generate the time key D_t, then returns it to $\mathcal{A}_{re}$.
  Secret Value query(ID): The challenger runs Set-Secret-Value to generate s_{ID}, then returns it to $\mathcal{A}_{re}$.
  Public Key request(ID): The challenger runs Set-Public-Key to generate the public key PK_{ID}. It returns PK_{ID} to $\mathcal{A}_{re}$.
  Signature query(M, ID, t): The challenger responds with user ID's signature on message M at the time period t.
- **Forge**: At the end of the game, $\mathcal{A}_{re}$ outputs a tuple $(S*, M*, ID*, t*)$ which means that it has forged a user $ID*$'s signature $S*$ on a message $M*$ at time period $t*$. $(S*, M*, ID*, t*)$ should not be an output of the signature oracle.

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

C. Complexity Assumption

We will use bilinear pairings in the concrete construction, and the security is based on the Computational Diffie-Hellman problem. So, in this section, we review the definition of a bilinear pairing and describe the Computational Diffie-Hellman assumption.

Bilinear Pairing. Suppose $\mathbb{G}_1$ is an additive cyclic group and $\mathbb{G}_2$ is a multiplicative cyclic group with the same prime order p. Let P denote a generator of $\mathbb{G}_1$. A map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ with the following properties is called a *bilinear pairing*:

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

Computational Diffie-Hellman (CDH) problem. *Given (aP, bP) with random $a, b \in \mathbb{Z}_p^*$, to compute abP.*

The Computational Diffie-Hellman assumption states that the CDH problem is hard.

III. A Revocable Certificateless Signature Scheme

This section gives the concrete construction of our revocable certificateless signature scheme.

- Setup: G_1 and G_2 are two cyclic groups of prime order p. P is a generator of G_1, and $e : G_1 \times G_1 \longrightarrow G_2$ 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 $(p, G_1, G_2, P, e, P_0, H_1, H_2, H_3, H_4)$ and the master secret key is s.
- Extract-Initial-Partial-Private-Key: Taking as input 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 private channel.
- Update-Time-Key: Taking as input an identity ID and a time period t, this algorithm computes $Q_{IDt} = H_2(ID, t)$ and the time key $D_{IDt} = sQ_{IDt}$, then transmits D_{IDt} to the user via a public channel.
- Set-Secret-Value: This algorithm produces a secret value $x_{ID} \in Z_p^*$ for the user ID.
- Set-Private-Key: For a user with identity ID at the time period t, the full private key SK_{IDt} is expressed as $(D_{ID} + D_{IDt}, x_{ID})$.
- 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 period 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 $V = D_{ID} + D_{IDt} + rH_3(M, ID, t, PK_{ID}, U) + x_{ID}H_4(M, ID, t, PK_{ID})$.
  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 $e(V, P) = e(Q_{ID} + Q_{IDt}, P_0)e(H_3(M, ID, t, PK_{ID}, U), U)$ $e(H_4(M, ID, t, PK_{ID}), PK_{ID})$ holds. If yes, output "accept"; otherwise, output "reject".

IV. Security and Efficiency Analysis

A. Security Proof

We analyze the security of the above scheme.

Theorem 1 Suppose 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 probability $\epsilon' \geqslant \frac{1}{q_2}\epsilon$ 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 a scalar multiplication.

Proof. We show how an algorithm B, with an instance (P, aP, bP), to compute abP by interacting with the adversary A_I.

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

A_I may query initial partial private keys, time keys, secret values, public keys and signatures. Also, A_I can replace public keys and make queries to the random oracles. All pairs of query/answer are maintained in lists.

H_1 queries: When 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: The H_2 list contains tuples $(ID_i, t_j, Q_{ij}, h_{2ij}, z)$. z denotes the number of this query among all H_2 queries. When 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: When 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: When 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.

Next, we assume that A_I always makes the appropriate H_1 and H_2 queries before making other related queries.

Initial Partial Private Key Extraction queries: When receiving an initial partial private 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: When receiving a time key 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: At the end, 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 a scalar multiplication.

Theorem 2 Suppose 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. We show how an algorithm B, with an instance (P, aP, bP), to compute abP by interacting with the 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} may make some queries in this phase. All records of query/answer are maintained in lists.

H_1 queries: When 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: When 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: When 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: When 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.

Secret Value queries: When 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: When 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: When receiving a signature query on (M, ID, t), B does the following:

- 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: At the end, 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 is equivalent to

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

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 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. We show how an algorithm B, with an instance (P, aP, bP), to compute abP by interacting with the adversary A_{in}.

B provides A_{in} with the system parameters $(p, G_1, G_2, P, e, P_0 = aP, H_1, H_2, H_3, H_4)$. 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 query initial partial private keys, time keys, secret values, public keys and signatures, as well as the random oracles. All query/answers are recorded in lists.

H_1 queries: When 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: The H_2 list contains tuples of the form $(ID_i, t_j, Q_{ij}, h_{2ij}, f)$. 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: When 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: When 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.

Now, we assume that A_{in} always makes the appropriate H_1 and H_2 queries before making other related queries described as follows.

Initial Partial Private Key Extraction queries: When 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: When 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 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 4 Suppose H_i $(1 \leqslant i \leqslant 4)$ are random oracles, our RCLS scheme is EUF-CMA secure.

B. Performance evaluation

1) Implementation: How to choose elliptic curves to obtain efficient cryptographic schemes is suggested in some literature [20], [21]. Two factors must be considered: the group size l of the elliptic curve and the embedding degree d. For a security level of 1024-bit RSA, the result of $l \times d$ should be more than 1024. Most of pairing-based schemes are implemented on the Type A and Type D elliptic curves [21]. For our purpose, the implementation is on a Type A elliptic curve $y^2 = x^3 + x$, with $\mathbb{G}_1 = \mathbb{G}_2$, p being 160 bits, $d = 2$ and l being 512 bits.

The running time is obtained on AMD FX-8120 8 Duo CPU at 3.1GHZ frequency and an OS of fedora 19. It is an average time by running the scheme 100 times under the PBC library [21]. The expensive paring is fast on a Type A curve with running time 1.2445ms. We describe the running times consumed by different algorithms (EIPPK: Extract-Initial-Partial-Private-Key, UTK: Update-Time-Key) in our scheme in Table 1. From the table we see that it takes approximately 3.993ms (actually a BLS short signature) for EIPPK and UTK. To sign a message, it takes 8.5225ms. To verify a signature, the running time is 12.4015ms.

Table 1.
The running time of different algorithms in our scheme

algorithm	EIPPK	UTK	Sign	Verify
time	3.993ms	3.993ms	8.5225ms	12.4015ms

2) Comparison: As is seen, in our RCLS scheme, a user's private key contains not only an initial partial private key and a secret value but also a time key. Revocation mechanism is based on updating the time key, which is transmitted from KGC to the user over public channels. This property makes our new scheme more applicable in practice. In Table 1, we make a comparison of computational cost, ciphertext-length and revocation-type of our scheme with that of a trivial RCLS 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 to the user via a secret channel).

Table 2. Comparison

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

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

In the table, "revocation" denotes what kind of channel is employed for updating keys. **Secret** channel indicates both more computation cost and more bandwidth usage. Clearly, our RCLS scheme has better performance.

V. Extension

In this section, we extend our scheme to resist signing key exposure (also called decryption key exposure in [13]). This threat is first considered against revocable schemes by Seo et al in PKC'13. It captures a realistic attack that the short-term signing key may be leaked. It is natural to require the cryptosystem to be secure even if all different short-term signing keys are exposed. The algorithms of Setup, Extract-Initial-Partial-Private-Key, Update-Time-Key and Set-Secret-Value are identical to that of the above scheme. We use Set-Signing-Key instead of Set-Private-Key.

- Set-Signing-Key: At a time period t, a user with identity ID randomly selects $z \in Z_p^*$, computes $W_{IDt} = z(D_{ID} + D_{IDt})$, $W_0 = zP_0$ and $W_1 = zP$. The signing key SK_{IDt} is $(W_{IDt}, W_0, W_1, x_{ID})$.
- Set-Public-Key: Compute $PK_{ID} = x_{ID}P$ as the public key.
- Sign: This algorithm is run by a signer. It takes as input a message M, a time period t and the signing key SK_{IDt}, then does the following:
  1) Choose $r \in Z_p^*$ at random and compute $U = rP$.
  2) Compute $V = W_{IDt} + rH_3(M, ID, t, PK_{ID}, U) + x_{ID}H_4(M, ID, t, PK_{ID})$.
  3) Output the signature $\sigma = (U, V, W_0, W_1)$.
- Verify: This algorithm takes as input a message-signature pair $(M, \sigma = (U, V, W_0, W_1))$, a time period t and the signer's public key ID and PK_{ID}, then checks whether both $e(W_0, P) = e(W_1, P_0)$ and $e(V, P) = e(Q_{ID} + Q_t, W_0)e(H_3(M, ID, t, PK_{ID}, U), U)$ $e(H_4(M, ID, t, PK_{ID}), PK_{ID})$ hold. If yes, output "accept"; otherwise, output "reject".

VI. Conclusions

Revocation mechanism is indispensable to the application of public key cryptosystems. In this paper, we concentrate on revocation in certificateless public key cryptosystems. On one hand, we present an efficient construction of a revocable certificateless signature (RCLS) scheme. On the other hand, we extend the new scheme to be signing-key-exposure resistant. 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 suitable for practical applications such as cloud computing. 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. The security proofs confirm that our RCLS scheme is provably secure based on the standard CDH problem.

References

[1] S.S. Al-Riyami, Cryptographic Schemes Based on Elliptic Curve Pairings, PhD thesis, Royal Holloway, University of London, 2004.

[2] S.S. Al-Riyami, K.G. Paterson, Certificateless Public Key Cryptography, In Asiacrypt 2003, LNCS 2894, pp 452-473, 2003.

[3] D. Boneh, X. Ding, G. Tsudik, C. Wong, A method for fast revocaiton of public key certificates and security capabilities, In the 10th USENIX Security Symposium, USENIX, 2001.

[4] D. Boneh, M. Franklin, Identity-based Encryption from the Weil Pairing, In CRYPTO 2001, LNCS 2139, pp 213-229, 2001.

[5] A. Boldyreva, V. Goyal, V. Kumar, Identity-based encryption with efficient revocation, In CCS 2008, ACM, pp 417-426, 2008.

[6] H.S. Ju, D.Y. Kim, D.H. Lee, J. Lim, K. Chun, Efficient Revocation of Security Capability in Certificateless Public Key Cryptography, In KES 2005, LNCS 3682, pp 453-459, 2005.

[7] Y.R. Lee, On the Security of Two Certificateless Signature Schemes, In Intelligent and Soft Computing, volume 167, pp 695-702, 2012.

[8] L. Shen, F. Zhang, Y. Sun, Efficient Revocable Certificateless Encryption Secure in the Standard Model, The Computer Journal (2013) doi: 10.1093/comjnl/bxt040. First published online: April 30, 2013.

[9] B. Libert, J.J. Quisquater, Efficient revocation and threshold pairing based cryptosystems, Symposium on Principles of Distributed Computing-PODC 2003, 2003.

[10] B. Libert, D. Vergnaud, Adaptive-ID secure revocable identity-based encryption, In CT-RSA 2009, LNCS 5473, pp 1-15, 2009.

[11] S. Micali, Novomodo: Scalable Certificate Validation and Simplified PKI Management, In the 1st Annual PKI Research Workshop 2002, pp 15-25, 2002.

[12] M. Myers, R. Ankney, A. Alpani, S. Galperin, C. Adams, X.509 Internet Public Key Infrastructure: Online Certificate Status Protocol (OCSP), RFC 2560.

[13] J.H. Seo, K. Emura, Revocable identity-based encryption revisited: security model and construction, In PKC 2013, LNCS 7778, pp 216-234, 2013.

[14] A. Shamir, Identity-based cryptosystems and signature schemes, In CRYPTO 1984, pp 47-53, 1984.

[15] S.S.M. Chow, C. Boyd, J.M.G. Nieto, Security-Mediated Certificateless Cryptography, In PKC 2006, LNCS 3958, pp 508-524, 2006.

[16] T.T. Tsai, Y.M. Tseng, T.Y. Wu, Revocable ID-based Signature Scheme with Batch Verifications, 2012 Eighth International Conference on Intelligent Information Hiding and Multimedia Signal, pp 49-54, July 2012.

[17] Y.M. Tseng, T.T. Tasi, Efficient revocable ID-based encryption with a public channel, The Computer Journal 2012, 55(4): 475-486.

[18] W.S. Yap, S.S.M. Chow, S.H. Heng, B.M. Goi1, Security Mediated Certificateless Signatures, In ACNS 2007, LNCS 4521, pp 459-477, 2007.

[19] Z. Zhang, D.S. Wong, J. Xu, D. Feng, Certificateless Public-Key Signature: Security Model and Efficient Construction, In ACNS 2006, LNCS 3989, pp 293-308, 2006.

[20] National Institute of Standards and Technology, Recommended elliptic curves for federal government use, July, http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf.

[21] PBC library, the pairing-based cryptography library, http://crypto.stanford.edu/pbc/.

Yinxia Sun received her B.S. and M.S. degree in mathematics from Nanjing Normal University, China, in 2003 and 2008, respectively, and Ph.D. degree in cryptography from Xidian University, China in 2011. She is a lecturer at School of Computer Science and Technology, Nanjing Normal University, China. Her research interests include public key cryptography and network security.

Futai Zhang is a professor at the School of Computer Science and Technology, NNU, China. His research interests include information security, network security and cryptography.

Limin Shen received the B.S. and M.S. degree in mathematics from Wuhan University, China, in 2001 and 2004, respectively. She is a lecturer at the School of Computer Science and Technology, NNU, China. Her research interests include information theory and coding, cryptography.