Proxy Re-signature Scheme Based on Quadratic Residues

Deng Yuqiao
Guangdong University of Business Studies, Mathematics and Computer Science College, Guangzhou 510640
Email: dengyuqiao80@yahoo.cn

Song Ge
Harbin Institute of Technology Shenzhen Graduate School, Shenzhen 150001
Email: carroll0708@qq.com

Abstract—In 1998, Blaze, Bleumer, and Strauss (BBS) proposed proxy re-signatures, in which a semi trusted proxy acts as a translator between Alice and Bob. The proxy re-signature schemes can be applied in many applications. However, the existing proxy re-signature schemes were all based on Diffie-Hellman assumption. In this paper, we present a proxy re-signature scheme based on quadratic residues, which is Bidirectional. The scheme is safety under the random oracle model. At the same time, the definition of forward-secure proxy re-signature is given, then a corresponding scheme is presented which can maintain the former signature’s safety even if current cycle’s Signature key has been leaked.

Index Terms—quadratic residues, proxy re-signature, forward-secure, random oracle model

I. INTRODUCTION

Proxy re-signature schemes, introduced by Blaze, Bleumer, and Strauss [1], and formalized later by Ateniese and Hohenberger [2]. Digital signature schemes allow a signer to transform any message into a signed message, such that anyone can verify the validity of the signed message using the signer’s public key, but only the signer can generate signed messages. In a proxy re-signature scheme, a semi-trusted proxy is given some information which allows it to transform Alice’s signature on a message \( m \) into Bob’s signature on \( m \), but the proxy cannot, on its own, generate signatures for either Alice or Bob. A proxy re-signature scheme has eight desirable properties [2] as follows, though none of existing schemes satisfies all properties at the same time, see Table 1.

1. Unidirectional: In a unidirectional scheme, a re-signature key allows the proxy to transform A’s signature to B’s but not vice versa. In a bidirectional scheme, on the other hand, the re-signature key allows the proxy to transform A’s signature to B’s as well as B’s signature to A’s.

2. Multi-use: A transformed signature can be re-transformed again by the proxy.

3. Private Proxy: The re-signature key can be kept secret by the proxy.

4. Transparent: A signature on the same message signed by the delegator is computationally indistinguishable from a signature transformed by a proxy.

5. Key-Optimal: In a key-optimal scheme, a user is required to protect and store only a small constant amount of secrets no matter how many signature delegations the user gives or accepts.

6. Non-interactive: The delegatee is not required to participate in a delegation process.

7. Non-transitive: A re-signing right cannot be redelegated by the proxy alone.

8. Temporary: A re-signing right is temporary.

The proxy re-signature schemes can be applied in many applications, for instance, we can use proxy re-signature schemes to simplify key management [1], provide proofs for a path that has been taken, manage group signatures, simplify certificate management [2], construct a Digital Rights Management (DRM) interoperable system [3].

Up to now, the security proven of existing proxy re-signature schemes is under Diffie-Hellman assumption [1,2,6,7,8,9]. In this paper, we first propose a proxy re-signature scheme based on quadratic residues, it satisfies bidirectional, multi-use, private proxy, transparent

<table>
<thead>
<tr>
<th>Property</th>
<th>BBS(2)</th>
<th>( S_{bs}^{[1]} )</th>
<th>( S_{as}^{[1]} )</th>
<th>Our Scheme 1</th>
<th>Our Scheme 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>8</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

TABLE I. THE PROPERTIES THAT SOME PROXY RE-SIGNATURE SCHEMES AND OURS SATISFY.
properties. We prove that it can resist adaptive chosen message attack.

Exposure of secret keys can be a devastating attack on a digital signature scheme since such an attack typically implies that all security guarantees are lost. The notion of forward security was recently proposed by Anderson [4] and later formalized by Bellare and Miner [5]. Based on the framework of the first scheme, we propose a forward secure proxy re-signature scheme, which can be considered as a forward secure extension of the first scheme.

II. BACKGROUND

A. quadratic residue

In number theory, an integer \( q \) is called a quadratic residue modulo \( n \) if it is congruent to a perfect square (mod \( n \)); i.e., if there exists an integer \( x \) such that:

\[
x^2 \equiv q \pmod{n}
\]

Otherwise, \( q \) is called a quadratic nonresidue (mod \( n \)).

Originally an abstract mathematical concept from the branch of number theory known as modular arithmetic, quadratic residues are now used in applications ranging from acoustical engineering to cryptography and the factoring of large numbers.

B. Remarks on the BBS Scheme

The authors of the BBS paper proposed some potential applications of proxy re-signatures. By taking a careful look at how one can use these proxy re-signatures in practice, we can see that the BBS construction has its own limitations. In brief, their scheme is actually “proxyless” since it is possible to recover the information that would be stored at the proxy by looking at the original signature and its transformation. This precludes the possibility of having a proxy in the first place since anyone would be able to impersonate the proxy itself once a single re-signature is released.

C. Forward secure signatures

The goal of forward security is to protect the risk of key exposure, but in a simple way, in particular without requiring distribution or protected storage devices, and without increasing key management costs.

A user begins, as usual, by registering a public key \( \text{pk} \) and keeping private the corresponding secret key, which we denote \( \text{sk}_0 \). The time during which the public key \( \text{pk} \) is desired to be valid is divided into periods, say \( T \) of them.

While the public key stays fixed, the user “evolves” the secret key with time. Thus in each period, the user produces signatures using a different signing key: \( \text{sk}_1 \), in period 1, \( \text{sk}_2 \), in period 2, and so on. The secret key in period \( i \) is derived as a function of the one in the previous period. The key evolution paradigm is illustrated in Figure 1.

III. DEFINITIONS

A. Bidirectional Proxy Re-Signature

Definition 1: A proxy re-signature scheme is a tuple of polynomial time algorithms \((\text{KeyGen}, \text{ReKey}, \text{Sign}, \text{ReSign}, \text{Verify})\), where:

- \((\text{KeyGen}, \text{Sign}, \text{Verify})\) form the standard key generation, signing, and verification algorithms.
- On input \((\text{sk}_1, \text{sk}_2)\), the re-signature key generation algorithm, \text{ReKey}, outputs a key \( \text{rk}_{s-a,b} \) for the proxy.
- On input \( \text{rk}_{s-a,b} \), a public key \( \text{pk}_d \), a message \( m \) and a signature \( \sigma_s(m) \), the re-signature function, \text{ReSign}, outputs B’s signature \( \sigma_s(m) \) if \( \text{Verify}(\text{pk}_d, m, \sigma_s(m)) \) and \( \bot \) otherwise.
- Correctness. For any message \( m \) in the message space and any key pairs 

\[
(\text{pk}, \text{sk}), (\text{pk}', \text{sk}') \leftarrow \text{KeyGen}(t^*)
\]

let
\[
\sigma = \text{Sign} (\text{sk}, m) \text{ and } \text{rk} = \text{ReKey}(\text{sk}, \text{sk}')
\]

then the following two conditions must hold:

\[
\text{Verify} (\text{pk}, m, \sigma) = 1 \quad \text{and} \quad \text{Verify}(\text{pk}', m, \text{ReSign}(\text{rk}, \text{pk}, m, \sigma)) = 1
\]

We define security for bidirectional proxy re-signature schemes by the following game between a challenger and an adversary: (Note that we adopt the method in [6] to define the security notion of bidirectional proxy re-encryption schemes: static corruption, i.e., in this security notion, the adversary has to determine the corrupted parties before the computation starts, and it does not allow adaptive corruption of proxies between corrupted and uncorrupted parties.)

- Queries. The adversary adaptively makes a number of different queries to the challenger. Each query can be one of the following.

\[
\text{Uncorrupted Key Generation } \text{O}_{\text{KeyGen}} : \text{Obtain a new key pair as } (\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(t^*)
\]

The adversary is given \( \text{pk} \).
● Corrupted Key Generation $O_{\text{KeyGen}}$: Obtain a new key pair as $(pk, sk) \leftarrow \text{KeyGen}(1^t)$. The adversary is given $pk$ and $sk$.

● Hash queries $O_h$: On input a message $m$ and a $R \in Z$, return $h = H(m, R)$.

● Re-Signature key Generation $O_{\text{ReKey}}$: On input $(pk, pk')$ by the adversary, where $pk, pk'$ were generated before by KeyGen, return the re-signature key $rk_{pk,pk'} = \text{Re Key}(sk, sk')$, where $sk, sk'$ are the secret keys that correspond to $pk, pk'$. Here, we also require that both $pk, pk'$ are corrupted, or both are uncorrupted.

● Re-signature $O_{\text{ReSign}}$: On input $(pk, pk', m, \sigma)$, where $pk, pk'$ were generated before by KeyGen. The adversary is given the re-signed signature $\sigma' = \text{Re Sign}(\text{Re Key}(sk, sk'), pk, m, \sigma)$, where $sk, sk'$ are the secret keys that correspond to $pk, pk'$.

● Signature $O_{\text{Sign}}$: On input a public key $pk$, a message $m$, where $pk$ was generated before by KeyGen. The adversary is given the corresponding signature $\sigma = \text{Sign}(sk, m)$, where $sk$ is the secret key that correspond to $pk$.

● Forgery. The adversary outputs a message $m^*$, a public key $pk^*$, and a string $\sigma^*$. The adversary succeeds if the following hold true:
  1. $\text{Verify}(pk^*, m^*, \sigma^*) = 1$.
  2. $pk^*$ is not from $O_{\text{KeyGen}}$.
  3. $(pk^*, m^*)$ is not a query to $O_{\text{Sign}}$.
  4. $(\varnothing, pk^*, m^*, \Delta)$ is not a query to $O_{\text{ReSign}}$, where $\varnothing$ denotes any public key, and $\Delta$ denotes any signature.

The advantage of an adversary $A$ in the above game is defined to be:

$$\text{Adv}_A = \text{Pr}[A \text{ succeeds}]$$

Where the probability is taken over all coin tosses made by the challenger and the adversary.

B. Bidirectional forward-secure Proxy Re-Signature

Definition 2: A forward-secure proxy re-signature scheme is a tuple of polynomial time algorithms $(\text{KeyGen}, \text{ReKey}, \text{Update}, \text{Sign}, \text{ReSign}, \text{Verify})$, where:

● The key generation algorithm, $\text{KeyGen}$, takes as input a security parameter $1^t$ and the total number of time periods $N$. It returns a public key $pk$ and an initial secret key $sk_i$.

● The key update algorithm, $\text{Update}$, takes as input a secret key $sk_{i-1}$ as well as the index $i$ of the current time period. It returns a secret key $sk_i$ for period $i$. (Note that, the key for the proxy also uses the Update algorithm to update its proxy key.)

● On input an key pair $(sk_i, sk_{i'})$, the re-signature key generation algorithm, $\text{ReKey}$, outputs an key $rk_{sk_i, sk_{i'}}$ for the proxy.

● On input an $ith$ secret key of the current time period $sk_i$, a message $m$, the sign algorithm, $\text{Sign}$, outputs signature $\sigma(m)$.

● On input an $ith$ re-sign key $rk_i$ of the current time period, public key pair $pk_i, pk_k$, a message $m$ and a signature $\sigma_i(m)$, the re-signature function, $\text{ReSign}$, outputs $B$’s signature $\sigma_i(m)$ if $\text{Verify}(pk_i, m, i, \sigma_i(m)) = 1$ and $\perp$ otherwise.

● On input a public key $pk$, a message $m$, an index $i$ of the current time period, a signature $\sigma$, the verify algorithm, $\text{Verify}$, outputs $1$ if $\sigma(m) = \text{sign}(sk_i, m)$ and $0$ otherwise.

● Correctness. For any message $m$ in the message space and any $ith$ key pairs of the current period: $(pk, sk_i), (pk', sk_{i'})$, let:
  $\sigma = \text{Sign}(sk_i, m)$ and $rk_i = \text{Re Key}(sk_i, sk_{i'})$.

Then the following two conditions must hold:

$$\text{Verify}(pk, m, i, \sigma) = 1\quad \text{Verify}(pk', m, i, \text{Re Sign}(rk_i, pk, pk', m, \sigma)) = 1$$

We also define the security notion of bidirectional forward-secure proxy re-signature with static corruption by a game between a challenger and an adversary.

● Queries. The adversary adaptively makes a number of different queries to the challenger. Each query can be one of the following.

● Uncorrupted Key Generation $O_{\text{KeyGen}}$: Obtain a new initial key pair as $(pk, sk_0) \leftarrow \text{KeyGen}(1^t)$. The adversary is given $pk$.

● Corrupted Key Generation $O_{\text{KeyGen}}$: Obtain a new key pair as $(pk, sk_0) \leftarrow \text{KeyGen}(1^t)$. The adversary is given $pk$ and $sk_0$.

● Hash queries $O_h$: On input a message $m$ and a $R \in Z$, return $h = H(m, R)$.

● Re-Signature key Generation $O_{\text{ReKey}}$: On input initial public key $(pk, pk')$ by the adversary and an index $i$ of current time period, where $(pk, pk')$ were generated before by KeyGen, return the $ith$ re-signature key $rk_{pk,pk'} = \text{Re Key}(sk_i, sk_{i'})$. Here, we also require that both $pk, pk'$ are corrupted, or both are uncorrupted.
Re-signature $O_{\text{ReSign}}$ : On input $(i, pk, pk', m, \sigma)$, where $pk, pk'$ were generated before by KeyGen, $i$ is the index of the current time period. The adversary is given the re-signed signature $\sigma' = \text{ReSign}(\text{ReKey}(sk, sk'), pk, pk', m, \sigma)$, where $sk, sk'$ are the $i$th keys that correspond to $pk$ at current time period.

- Signature $O_{\text{Sign}}$ : On input a public key $pk$, a message $m$ and a time period $i$, where $pk$ was generated before by KeyGen. The adversary is given the corresponding signature $\sigma = \text{Sign}(sk, m)$, where $sk$ is the secret key that correspond to $pk$ at $i$th time period.

- Forgery. The adversary outputs a message $m^*$, a public key $pk^*$, an index $i^*$ of current time period, and a string $\sigma^*$. The adversary succeeds if the following hold true:
  1. $\text{Verify}(pk^*, m^*, i^*, \sigma^*) = 1$.
  2. $(pk, m^*, i^*)$ is not from $O_{\text{KeyGen}}$.
  3. $(pk^*, m^*, \sigma^*)$ is not a query to $O_{\text{Sign}}$.
  4. $(i^*, \sigma, \sigma^*, \Delta)$ is not a query to $O_{\text{ReSign}}$, where $\Delta$ denotes any signature.

The advantage of an adversary $A$ in the above game is defined to be $\text{Adv}_A = \text{Pr}[A \text{ succeeds}]$, where the probability is taken over all coin tosses made by the challenger and the adversary.

IV. BIDIRECTIONAL PROXY RE-SIGNATURE SCHEMES

A. Bidirectional Proxy Re-signature Schemes Based on quadratic residues

- KeyGen: On input the security parameter $k$, it chooses two security prime numbers $p = 2p_1 + 1$, $q = 2q_1 + 1$, where $p_1, q_1$ are also prime numbers. Let $n = p \times q$ and $\log_2 n > k$. It chooses a security hash function $H : \{0, 1\}^* \rightarrow \{0, 1\}^k$. It selects a random $s \in \mathbb{Z}_n^*$, and output the key pair $pk = s^3 \mod n$, the public parameters $(k, n, p, q, H, pk)$.

- ReKey: On input two secret keys $sk_1 = s, sk_2 = t$, output the re-signature key $rk_{\text{Re}} = t / s$.

(Note that we make use of the same method in [2] to get the re-signature key)

- Sign: On input a secret key $sk = s$ and a message $m$, compute
  
  \[
  R = r^2 \mod n, h = H(m, R),
  \]

  output
  \[
  \sigma = (R, S) = (r^2 \mod n, r \times sk^3 \mod n).
  \]

- ReSign: On input a re-signature key $rk_{\text{Re}}$, a public key $pk$, a message $m$ and a signature $\sigma$, check that $\text{Verify}(pk, m, \sigma) = 1$. If $\sigma$ does not verify, output $\bot$; otherwise, output $\sigma' = (R', S') = (R, rk_{\text{Re}})^{(m,R)} \times S$.

Verify: On input a public key $pk$, a message $m$, and a purported signature $\sigma = (R, S)$, output $1$, if $S = R \times pk^{(m, R)}$ and 0 otherwise.

B. Security Analysis

Lemma 1[10] Let $G$ be a group, $e_1, e_2 \in \mathbb{Z}_p$, $\text{gcd}(e_1, e_2) = 1$, given $a, b \in G$, we have $a^{e_1} = b^{e_2}$, then we can compute $e \in G$, where $c = a \text{ or } c = b$.

Theorem 1 (Security) In the random oracle model, the bidirectional proxy re-signature scheme we presented is correct and existentially unforgeable under the lemma 1.

Proof. The correctness property is easily observable. We show security using forking lemma [11].

If there exists an adversary $A$ that can break the above proxy re-signature scheme with non-negligible probability $\varepsilon$ after making at most $d$ sign queries, $q_S$ resign queries, $q_K$ (un)corrupted key queries and $q_H$ hash queries, then there also exists an adversary $B$ that can solve the quadratic residues problem in $G$ with probability $\varepsilon' \geq \frac{1}{2d^4} (1 - \frac{d_1}{2^4})^2 \varepsilon$.

On input $x^2 = y \in QR_n$, it is difficult to calculate the quadratic residues of $y$, that is, $x$. Quadratic residues adversary $B$ simulates a bidirectional proxy re-signature security game for $A$ as follows:

- Queries: $B$ builds the following oracles:
  \[
  O_{\text{KeyGen}}: B \text{ chooses a random } x_i \in \mathbb{Z}_n^*, \text{ and outputs } pk_i = x_i^3 \mod n, \text{ the public parameters } (k, n, p, q, H, pk_i).
  \]
  \[
  O_{\text{ReKey}}: B \text{ chooses a random } x_i \in \mathbb{Z}_n^*, \text{ and outputs } (pk_i, sk_i) = (x_i^3, x_i).
  \]

- $O_{H}$: $B$ maintains a hash table $\{(m, r_{a_i}, R_{a_i}, h_{a_i})\}$, we noted it as $T_H$. For each query to $O_H$ on input $m$, check if there is an entry in $T_H$. If so, output the corresponding value, otherwise $B$ chooses a random $r_{a_i} \in \mathbb{Z}_n^*$, let $R_{a_i} = r_{a_i}^3 \mod n$, then $B$ chooses a random $h_{a_i} \in \{0, 1\}^k$ and output $h_{a_i}$ as the corresponding value. Record the pair $(m, r_{a_i}, R_{a_i}, h_{a_i})$ in table $T_H$.

- $O_{\text{Sign}}$: On input $(pk_i, m)$, if $pk_i$ is corrupted, $B$ returns the signature $\sigma = (R, S) = (r^2 \mod n, r \times sk_i^3 \mod n)$, where $h = H(m, R)$. Otherwise, $B$ performs as follows.

- $B$ maintains a signature table $\{(m, r_{a_i}, R_{a_i}, S_{a_i})\}$, where $i = 1, 2, \ldots, q_S$, we noted it as $T_S_i$. If $(m, a_i, b_i) \in T_S_i$, $B$ returns $S_{a_i}$; otherwise, $B$ chooses $a_i, b_i \in \mathbb{Z}_n^*$, let $r_{a_i} = y^{a_i} \mod n$, $h_{a_i} = 2b_i$, $R_{a_i} = y^{2b_i} \mod n$.
if \((m_1, \bullet, R_m, \bullet) \in T_H\), \(B\) outputs \(\bot\). The probability of this event is \(q_H^2\), otherwise, computes \(S_m = y^{m-1}\) and output \(\sigma = (R_m, S_m)\). Record the pair \((m_1, r_m, R_m, S_m)\) in table \(T_2\).

\(O_{\text{Re}, \text{Key}}\) : On input \((pk_1, pk_j)\), if \(pk_1\) and \(pk_j\) are both corrupted or both uncorrupted, \(B\) returns \(r_{k_{i,j}} = (x_i / x_j) \text{mod } p\); else, this input is illegal.

\(O_{\text{ReSign}} : \) On input \((rk_{i,j}, pk, m, \sigma)\). If \(\text{Verify}(pk, m, \sigma) \neq 1\), \(B\) outputs \(\bot\). Otherwise, \(B\) does:

If \(pk_1\) and \(pk_j\) are both corrupted or both uncorrupted, output \(\text{ReSign}(O_{\text{Re}, \text{Key}}, (pk_1, pk_j), pk, m, \sigma)\).

Else, \(B\) output \(\text{ReSign}(pk, m, \sigma)\).

\(B\) runs the algorithm for \(1/\varepsilon\) times, according to the assumption, we can get a signature \(\sigma = (R_m, S_m)\).

Run the above algorithm twice. In each run, \(B\) queries the random oracle for a sequence of new input, then \(B\) will get two signatures: \(\sigma = (R_m, S_m)\) and \(\sigma' = (R_m', S_m')\).

According to forking lemma, if \(m^* = m^{*'}\), then we can get \(r_n, r_n', R_n, R_n'\) from \(H\) table, where:

\[
R_m = r_n^2 \text{mod } n
\]  
\[
R_m' = r_n'^2 \text{mod } n
\]
\[
S_m^2 = R_m \times y^{k_m}
\]
\[
S_m'^2 = R_m' \times y^{k_m'}
\]
\[
y^{(r_n - k_m)} = \frac{S_m}{S_m'} \left(\frac{r_n}{r_n'}\right)
\]

If \(\gcd(h_m - h_m', 2) = 1\), according to lemma 1, we can find the quadratic residues of \(y\).

The probability that the signature pair \((m_1, \bullet, R_m, \bullet)\) not in \(H\) table is \(1 - \frac{q_H^2}{2}\), the probability that \(\gcd(h_m - h_m', 2) = 1\) holds is \(1/2\). In forking lemma, the probability of \(m^* = m^{*'}\) holds is \(\frac{1}{q_H}\), so, \(B\) can solve the Quadratic Residues problem with probability \(\varepsilon' \geq \frac{1}{2}q_H^2\frac{1}{2}(\frac{q_H}{2})^n\).

IV. A FORWARD SECURE PROXY RE-SIGNATURE SCHEME

Based on the above proxy re-signature scheme, we present a forward secure proxy re-signature scheme as follows. The scheme’s security is base on the Strong RSA Assumption[9].

A. Forward Secure Proxy Re-signature Scheme

KeyGen: On input the security parameter \(k\), it chooses two security prime numbers \(p = 2p_i + 1\), \(q = 2q_i + 1\), where \(p_i, q_i\) are also prime numbers. Let \(n = p \times q\) and \(\log, n > k\). It chooses a security hash function \(H: \{0,1\}^* \rightarrow \{0,1\}^k\). It selects a random \(s \in Z_n^\times\) and a total time interval number \(T\), output the key pair \(pk = s^{\alpha_1} \text{mod } n\), the public parameters \((k, n, p, q, H, T, pk)\).

ReSign: On input two secret keys \(sk_i = s^{\alpha'_i}\) for period \(i\), output the re-signature key \(rk_{i \rightarrow B} = (t / s)^{\alpha_i}\).

Update: On input secret key \(sk_{i-1}\) for period \(i - 1\), output \(sk_i = sk_{i-1}^2\) for period \(i\).

Sign: On input a secret key \(sk = s^{\alpha_i}\) for period \(i\) and a message \(m\), compute \(R = r^{\alpha_i} \text{mod } n\), \(h = H(m, R)\), output \(\sigma = (R, S) = (r^{\alpha_i} \text{mod } n, S = r \times s^{k_{i-1}} \text{mod } n}\).

ReSign: On input a re-signature key \(rk_{i \rightarrow B}\) for period \(i\), a public key \(pk_i\), a message \(m\) and a signature \(\sigma_i\), check that \(\text{Verify}(pk_i, m, \sigma_i) = 1\). If \(\sigma_i\) does not verify, output \(\bot\); otherwise, output \(\sigma' = (R', S') = (R, rk_{i \rightarrow B}^{H(m, R)} \times S)\).

Verify: On input a public key \(pk\), a message \(m\), and a purported signature \(\sigma = (R, S)\), output \(1\), if \(S^{\alpha_i} = R_x pk^{H(m, R)}\) and \(0\) otherwise.

B. Forward Security

The correctness property is easily observable. If an adversary gets a secret key \(sk_j\) for period \(i\), he can not compute a secret key \(sk_i (j < i)\), otherwise, he can compute a number \(c \in Z_n^\times\) where \(c^{2^j} \text{mod } n = sk_i\), that is, he can solve the Strong RSA Assumption. Thus, the scheme is forward secure. The scheme is also existentially unforgivable.

B. Security Analysis

Theorem 2 (Security) In the random oracle model, the forward secure proxy re-signature scheme we presented is correct and existentially unforgivable under the lemma 1.

Proof. The correctness property is easily observable. We show security using forking lemma [11].

If there exists an adversary \(A\) that can break the above proxy re-signature scheme with non-negligible probability \(\varepsilon\) after making at most \(q_\text{sign}\) sign queries, \(q_\text{resign}\) resign queries, \(q_\text{K}\) (un)corrupted key queries and \(q_H\) hash queries, then there also exists an adversary \(B\) that can solve the quadratic residues problem in \(G\) with probability \(\varepsilon' \geq \frac{1}{2}q_H^2\frac{1}{2}(\frac{q_H}{2})^n\).
On input \( e^{2i\pi} = \alpha \mod n \), it is difficult to calculate \( e \in \mathbb{Z}_n^* \) \( \epsilon \). wee adversary \( B \) simulates a bidirectional proxy re-signature security game for \( A \) as follows:

Queries: \( B \) builds the following oracles:

- **O_{KeyGen}**: \( B \) chooses a random \( x_i \in \mathbb{Z}_n^* \), and outputs \( pk_i = x_i^2 \mod n \).
- **O_{KeyGen}**: \( B \) chooses a random \( x_i \in \mathbb{Z}_n^* \), and outputs \( (pk_i, sk_i) = (x_i^2, x_i) \).

- **O_{Hi}**: \( B \) maintains a hash table \( \{(m_i, r_{m_i}, R_{m_i}, h_{m_i})\} \), we noted it as \( T_{H_i} \). For each query to \( O_{Hi} \) on input \( m_i \), check if there is an entry in \( T_{H_i} \). If so, output the corresponding value, otherwise \( B \) chooses a random \( r_{m_i} \in \mathbb{Z}_n^* \), and outputs \( (pk_i, sk_i) = (r_{m_i}^2, r_{m_i}) \mod n \), then \( B \) chooses a random \( h_{m_i} \in \{0,1\}^q \) and output \( h_{m_i} \) as the corresponding value. Record the pair \( (m_i, r_{m_i}, R_{m_i}, h_{m_i}) \) in \( T_{H_i} \).

- **O_{Sig}**: On input \( (pk_i, m) \), if \( pk_i \) is corrupted, \( B \) returns the signature \( \sigma = (R, S) = (\nu \mod n, r \times s_k^i \mod n) \), where \( h = H(m, R) \). Otherwise, \( B \) performs as follows.

\( B \) maintains a signature table \( \{(m_i, r_{m_i}, R_{m_i}, S_{m_i})\} \), where \( i = 1, 2, ..., q_s \), we noted it as \( T_S \). If \( (m_i, \bullet, \bullet, \bullet) \in T_S \), \( B \) returns \( S_{m_i} \); otherwise, \( B \) chooses \( a_i, b_i \in \mathbb{Z}_n^* \), let \( r_{m_i} = y^{a_i} \mod n \), \( h_{m_i} = 2^{i+1} b_i \), \( R_{m_i} = y^{2^{i+1} \cdot a_i} \mod n \), if \( (m_i, \bullet, R_{m_i}, \bullet) \in T_H \), \( B \) outputs \( \perp \). The probability of this event is \( \frac{q_H}{2^q} \), otherwise, computes \( S_{m_i} = r_{m_i}^{a_i} \) and output \( \sigma = (R_{m_i}, S_{m_i}) \). Record the pair \( (m_i, r_{m_i}, R_{m_i}, S_{m_i}) \) in table \( T_S \).

- **O_{RxSig}**: On input \( (pk_i, pk_j) \), if \( pk_i \) and \( pk_j \) are both corrupted or both uncorrupted, \( B \) returns \( r_{k_{i,j}} = (x_i / x_j) \mod p \); else, this input is illegal.

- **O_{RxSign}**: On input \( (r_{k_{i,j}}, pk_i, m, \sigma) \). If \( \text{Verify}(pk_i, m, \sigma) \neq 1 \), \( B \) outputs \( \perp \). Otherwise, \( B \) does:

\( B \) output the signature \( Sig(O_{RxSig}(pk_i, pk_j, m)) \).

- **O_{ReSig}**: On input \( (pk_i, m, \sigma) \). If \( \text{Verify}(pk_i, m, \sigma) \neq 1 \), \( B \) outputs \( \perp \). Otherwise, \( B \) does:

- **O_{ReSign}**: On input \( (pk_i, m) \).

\( B \) runs the algorithm for \( 1/\epsilon \) times, according to the assumption, we can get a signature \( \sigma = (R_{m_i}, S_{m_i}) \).

Run the above algorithm twice. In each run, \( B \) queries the random oracle for a sequence of new input, then \( B \) will get two signatures: \( \sigma = (R_{m_i}, S_{m_i}) \) and \( \sigma' = (R_{m_i'}, S_{m_i'}) \).

According to forking lemma, if \( m^* = m^{*'} \), then we can get \( r_{m^*}, r_{m^{*'}}, R_{m_i}, R_{m^{*}} \) from \( H \) table, where:

\[
R_{m^*} = r_{m^*}^{2^i \cdot a_i} \mod n \quad (10)
\]

\[
S_{m^*} = R_{m^*} \times y^{h_{m^*}} \quad (12)
\]

\[
S_{m^{*'}} = R_{m^{*'}} \times y^{h_{m^{*'}}} \quad (13)
\]

\[
(12) \doteq (13) \quad S_{m^{*'}} = \frac{r_{m^{*'}}^{2^i \cdot a_i}}{r_{m^*}^{2^i \cdot a_i}} \quad (14)
\]

If \( \gcd(h_{m^*} - h_{m^{*'}}, 2) = 1 \), according to lemma 1, we can find the quadratic residues of \( y \).

The probability that the signature pair \((m_i, \bullet, R_{m_i}, \bullet)\) not in \( H \) table is \( \frac{q_H}{2^q} \), the probability that \( \gcd(h_{m^*} - h_{m^{*'}}, 2) = 1 \) holds is \( \frac{1}{2} \). In forking lemma, the probability of \( m^* = m^{*'} \) holds is \( \frac{1}{2q_H} \), so, \( B \) can solve the Quadratic Residues problem with probability \( \epsilon' \geq \frac{1}{2q_H} (1 - \frac{q_H}{2^q}) \).

V. CONCLUSION

We have presented two proxy re-signature schemes based on quadratic residues problem which are proven secure in the random oracle model. Especially, the second one is an forward secure proxy re-signature scheme, which can greatly reduce the impact that the leakage of secret key.

REFERENCES


© 2011 ACADEMY PUBLISHER
Yuqiao Deng was born in China in 1980. He received the PhD degrees in south China University of technology, Guangdong, China, in 2010. He became a lecturer in Guangdong University of Business Studies, China, in 2010. His research interests include encryption, digital signature, cryptographic protocol and digital right management.

Ge Song was born in China in 1984. She is a doctoral student in Harbin Institute of Technology, Harbin, China, in 2010. Her research interests include data mining, short texts analysis, digital signature.