Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/090653

On-Line/Off-Line Threshold Proxy Re-Signature Scheme through the Simulation Approach

*Xiaodong Yang**, *Guojuan Gao*, *Yanan Li*, *Yan Li* and *Caifen Wang*

College of Computer Science and Engineering, Northwest Normal University, 730070, Gansu Lanzhou, China

Received: 27 Jun. 2014, Revised: 6 Dec. 2014, Accepted: 16 Dec. 2014
Published online: 1 Nov. 2015

Abstract: On-line/off-line threshold proxy re-signatures can efficiently improve the the performance of threshold proxy re-signature schemes. A simulation theorem for on-line/off-line threshold proxy re-signature schemes is presented in this paper. This theorem provides a theoretical basis for constructing an on-line/off-line threshold proxy re-signature scheme through the simulation approach, where the security of an on-line/off-line threshold proxy re-signature scheme can be reduced to that of its underlying divisible on-line/off-line proxy re-signature scheme. Furthermore, we propose an on-line/off-line threshold proxy re-signature scheme which is proven secure (unforgeable and robust) under the computational Diffie-Hellman assumption and the discrete logarithm assumption. The on-line phase of the proposed scheme is efficient: computing a re-signature does not require any heavy computations such as exponentiations or pairings.

Keywords: threshold proxy re-signature, divisible proxy re-signature, computational Diffie-Hellman problem, discrete logarithm problem

1 Introduction

Proxy re-signatures, introduced by Blaze *et al.* [1], enable a semi-trusted proxy to convert signatures from a delegatee into signatures from a delegator on the same messages by using a re-signing key. However, the proxy can not learn anything about either signing key. Due to the transformation function, proxy re-signature schemes can be used in many applications [2]. For example, in the case of an e-passport, the various check points can be associated with a semi-trusted proxy, which turns the signature of the traveler into the signature of the corresponding check point indicating which the traveler has been verified at the point. Proxy re-signature schemes can also be used in other applications, such as management of group signatures [3], identity authentication, and construction of digital rights management (DIM) interoperable system [4].

However, the semi-trusted problem is the serious defect in the existing proxy re-signature schemes [5,6,7]. Because the proxy has the re-signing key, he can transform signatures signed by the delegatee into signatures of the delegator. To address the semi-trusted issue, Yang *et al.* [8,9] produced the notion of threshold proxy re-signatures **TPRS** to avoid concentrating trust in any single proxy. In a threshold proxy re-signature scheme, given a set of n proxies and a threshold $t < n$, re-signatures can be generated by a set of at least $t+1$ proxies (rather than a single proxy) who hold the re-signature key in a shared form among them. A (t,n) threshold proxy re-signature scheme is said to be existential unforgeability, if no coalition of t proxies can not generate a new valid signature/re-signature on a new message even t proxies are corrupted by an adversary. It means that at most t of n proxies may be compromised without endangering the security of the threshold proxy re-signature scheme.

At present, the direct reduction approach and the simulation approach are two ways to prove existential unforgeability of a threshold proxy re-signature scheme [8,9]. In the direct reduction approach, the security of a threshold proxy re-signature scheme can directly reduced to the hardness of an underlying difficult problem such as the discrete logarithm problem. Based on a simulation theorem, the security of a threshold proxy re-signature scheme is directly reduced to the security of its underlying proxy re-signature scheme through the simulation approach. In fact, two approaches are essentially the same. The security of the threshold scheme

* Corresponding author e-mail: y200888@163.com

is eventually reduced to the hardness of an underlying difficult problem. However, if the underlying proxy re-signature scheme is unforgeable, the simulation approach can efficiently simplify the proof of the threshold proxy re-signature scheme.

The time to generate re-signatures is the biggest obstacle in the actual implementation of a threshold proxy re-signature scheme, since the costs of re-signature key operations required by a underlying proxy re-signature are amplified by the distributed protocols which produce re-signatures. Therefore, it is very important to seek methods of accelerating re-signature computation without compromising security.

To improve the performance of threshold proxy re-signature schemes, a notion called on-line/off-line threshold proxy re-signatures **OTPRS** is introduced in this paper. The idea is that the on-line/off-line techniques [10,11,12,13,14] are applied to a threshold proxy re-signature scheme in order to move most of re-signature computations to the off-line phase. In an on-line/off-line threshold proxy re-signature scheme, the proxies perform the bulk of re-signature computation in the off-line phase before knowing the message to be re-signed. The results of this pre-computation are stored and then utilized to re-sign actual messages in the on-line phase. Consequently, the computation of the authentic re-signature does not require any heavy computations such as exponentiations or pairings.

In order to efficiently construct on-line/off-line threshold proxy re-signature schemes through the simulation approach, we introduce a notion called divisible on-line/off-line proxy re-signatures **DOPRS** in this paper. A divisible on-line/off-line proxy re-signature scheme divides the re-signing computation into two stages: off-line (before knowing the message to be re-signed) and on-line (when the message must be re-signed). The re-signature of the message consists of two parts. The first part is called the off-line re-signature token, which is generated and may be sent to the recipient in the off-line stage. This part might be exposed before the message to be re-signed is given. The second part is called the on-line re-signature token, which is generated and may be sent to the recipient in the on-line stage.

In an on-line/off-line threshold proxy re-signature scheme, partial re-signature might be exposed all proxies in the off-line re-signing stage. Because the simulation theorem for ordinary threshold proxy re-signature schemes [8,9] implicitly assumes that no partial re-signature can be exposed in the re-signing stage, this theorem is not suitable for on-line/off-line threshold proxy re-signature schemes. We propose a simulation theorem for on-line/off-line threshold proxy re-signature schemes in this paper. This theorem shows that if an on-line/off-line threshold proxy re-signature scheme is simulatable from a divisible on-line/off-line proxy re-signature scheme, then the unforgeability of an on-line/off-line threshold proxy re-signature scheme can be reduced that of its underlying divisible on-line/off-line proxy re-signature scheme. Using this simulation theorem, we can efficiently build **OTPRS** schemes based on **DOPRS** schemes.

1.1 Related work

Yang *et al.* [9] proposed a bidirectional threshold proxy re-signature scheme through the direct reduction approach, which is existentially unforgeable and robust without random oracle. We denote this scheme by **BTPRS10**. Later, Yang *et al.* [8] proposed two threshold proxy re-signature schemes through the simulation approach. The first one, referred to as **BTPRS11**, is a bidirectional scheme which is existentially unforgeable and robust without random oracle. The second scheme, referred to as **UTPRS11**, is a unidirectional scheme which is existentially unforgeable and robust in the random oracle model. As far as we know, there exists no **OTPRS** scheme that has been publicly put forward yet.

1.2 Our contribution

In this paper, we first give a simulation theorem for on-line/off-line threshold proxy re-signature schemes (Theorem 1). This theorem provides a theoretical basis for efficiently constructing on-line/off-line threshold proxy re-signature schemes through the simulation approach. Based on Shao *et al.*'s proxy re-signature scheme $\mathbf{S_{mb}}$ [7], we then propose a divisible on-line/off-line proxy re-signature scheme which is existentially unforgeable without random oracle. Using the proposed divisible scheme as an ingredient, we present an efficient on-line/off-line threshold proxy re-signature scheme through the simulation approach, which is existentially unforgeable and robust under the computational Diffie-Hellman (CDH) and discrete logarithm assumptions. Our scheme can achieve robustness in the presence of $\lceil n/4 \rfloor$ malicious proxies. The on-line stage of the proposed scheme is efficient: computing a re-signature only requires a few modular multiplications when a message must be re-signed.

1.3 Organization

Our paper is organized as follows. We review some preliminaries and basic protocols in Section 2. The simulation theorem for **OTPRS** schemes is given in Section 3. Section 4 gives a **DOPRS** scheme. A **OTPRS** scheme is presented in Section 5 and our paper is concluded in Section 6.

2 Preliminaries and building blocks

2.1 Notations and definitions

We use $|x|$ to denote the length of x. If S is a finite set, by $s \in_R S$ means selecting an element s from S uniformly at random. We use $\mathbb{N}$ to denote the set of natural numbers, $\mathbb{R}$ to denote the set of real numbers, and $\mathbb{Z}$ to denote the set of integers. For $l \in \mathbb{N}$, we use 1^l to denote the concatenation of l ones, $\{0,1\}^*$ to denote the set of binary strings of arbitrary length, $dl_g h$ to denote the discrete logarithm of h to the base of g. PPT is the abbreviation of probabilistic polynomial-time.

Definition 1 A function $\eta : \mathbb{N} \to \mathbb{R}$ is negligible if for all constants $c > 0$, there exists an integer $N \in \mathbb{N}$ such that for all integers $k > N$ it holds that $\eta(k) < k^{-c}$.

Definition 2 Given a group G of prime order p and a generator g of G, the discrete logarithm problem in G is to compute $x \in \mathbb{Z}_p$ given (p, g, g^x). The discrete logarithm assumption in G states that no PPT algorithm solves the discrete logarithm problem with non-negligible probability.

Definition 3 Given a group G of prime order p and a generator g of G, the computational Diffie-Hellman (CDH) problem in G is to compute g^{ab} given $(g, g^a, g^b) \in G^3$ for some $a, b \in \mathbb{Z}_p$. The CDH assumption in G states that no PPT algorithm solves the CDH problem with non-negligible probability.

Definition 4 G_1 and G_2 are two multiplicative cyclic groups of prime order p, and g is a generator of G_1. A map $e : G_1 \times G_1 \to G_2$ is a bilinear paring with the following properties:

1. Bilinearity: $e(g^a, g^b) = e(g, g)^{ab}$, for all $a, b \in \mathbb{Z}_p$.
2. Non-degeneracy: $e(g, g) \neq 1_{G_2}$.
3. Computability: $e(g^a, g^b)$ can be computed in polynomial time.

2.2 Building blocks

We briefly review some basic protocols that we will use in our scheme. We assume that secrets of all protocols presented in this subsection are shared through Shamir's secret sharing scheme, using polynomials of degree t. We use $P_1, \cdots, P_n$ to denote n players participated in the following protocols.

Shamir's secret sharing scheme. By using this scheme, a dealer can distribute the secret among a group of n players in the following way [15].

– **Distribution phase**: Given a prime p and a shared secret $s \in \mathbb{Z}_p^*$, the dealer first chooses a number t satisfying $1 \leq t < n < p$. Then, the dealer chooses a_1, $\cdots$, $a_t \in \mathbb{Z}_p^*$, and constructs a polynomial $f(x)$ of degree t such that

$$f(x) = s + a_1 x + a_2 x^2 + \cdots + a_t x^t .$$

For $1 \leq i \leq n$, the dealer computes $s_i = f(i) \in \mathbb{Z}_p^*$ and sends s_i to the i-th player P_i.

– **Reconstruction phase**: Let $\Phi \subset \{P_1, \cdots, P_n\}$ be a set of players such that $|\Phi| > t$. These participants can reconstruct the polynomial $f(x)$ by computing $f(x) = \sum_{j \in \Phi} \lambda_{jx} s_j$, where $\lambda_{jx} = \prod_{l \in \Phi, l \neq j} \frac{x-l}{j-l} \in \mathbb{Z}_p$ is the Lagrange interpolation coefficient. Note that $s = f(0) = \sum_{j \in \Phi} \lambda_{j0} s_j$ and $\lambda_{j0} = \prod_{l \in \Phi, l \neq j} \frac{-l}{j-l}$.

If there are at most d malicious participants, by using error-correcting techniques [16], the number of players required to correctly reconstruct the secret s is at least $t + 2d + 1$ according to the Berlekamp-Welch bound. We use s=EC-Interplate$(s_1, \cdots, s_n)$ to denote an execution of this error-correcting-based reconstruction algorithm [16].

Multiplying two shared secrets (MUL). The multiplication protocol proposed by Ben-Or *et al.* [17] allows to multiply two shared secrets. Assume that player P_i holds the shares (a_i, b_i) of two secrets (a, b). The players jointly run this protocol to compute $c = ab$. Each player P_i receives his share c_i of c at the end of the protocol.

Computing shares of the inverse of a shared secret (INV). Given a prime p and a shared secret $a \in \mathbb{Z}_p^*$, the player P_i holds the shares a_i of a. The well-known protocol proposed by Bar-Ilan and Beaver [18] allows to compute shares of a^{-1}, such that $a \cdot a^{-1} \equiv 1 \bmod p$. Each player P_i can obtain his share a_i^{-1} of a^{-1} at the end of the protocol.

Distributed key generation protocol (DKG). Our scheme uses a discrete logarithm-based DKG protocol (see [19] for details) to jointly produce a random shared secret s (where the public value is g^s for some g). This protocol has the property that there is a simulator SIM_{DKG} which on input g^s can simulate the execution of an the adversary, such that the output is fixed to be g^s. Moreover, SIM_{DKG} can recover secret shares held by the corrupted participants.

2.3 Divisible on-line/off-line proxy re-signature

Definition 5 A divisible on-line/off-line proxy re-signature scheme **DOPRS** = (**KeyGen**, **ReKey**, **Sign**, **ReSign$^{\mathbf{off}}$**, **ReSign$^{\mathbf{on}}$**, **Verify**) is defined by the following algorithms:

– $(sk, pk) \leftarrow$ **KeyGen**(1^k) is the key generation algorithm. Given a security parameter $k \in \mathbb{N}$, this algorithm outputs signer's secret/public key pair (sk, pk).

– $RSK_{A \to B} \leftarrow$ **ReKey**(sk_A, sk_B, pk_A, pk_B) is the re-signature key generation algorithm. On input an (optional) delegatee's secret key sk_A, a delegator's secret key sk_B, and the corresponding public keys (pk_A, pk_B), this algorithm outputs a re-signature key $RSK_{A \to B}$ for the proxy to convert the delegatee's signatures into the delegator's signatures.

–$\sigma \leftarrow$ **Sign**(sk, m) is the signing algorithm. Given a secret key sk and a message m, it generates signature σ on m.
–$(\sigma_B^{off}, St) \leftarrow$ **ReSign**$^{\textbf{off}}(pk_A, pk_B, RSK_{A\to B})$ is the off-line re-signing algorithm. Given a re-signature key $RSK_{A\to B}$, a delegatee's public key pk_A and a delegator's public key pk_B, it generates a public off-line re-signature token σ_B^{off} and a secret state information St. This state information will be passed to the execution of the on-line re-singing algorithm.
–$\sigma_B^{on} \leftarrow$ **ReSign**$^{\textbf{on}}(RSK_{A\to B}, St, pk_A, m, \sigma_A)$ is the on-line re-signing algorithm. Given a re-signature key $RSK_{A\to B}$, a state information St, a delegatee's public key pk_A, a message m and a signature σ_A, it first validates that σ_A is valid under pk_A. It then generates an on-line re-signature token σ_B^{on} if **Verify**$(pk_A, m, \sigma_A) = 1$, $\perp$ otherwise. The re-signature for m is defined as $\sigma_B = (\sigma_B^{off}, \sigma_B^{on})$ which verifies under the delegator's public key pk_B.
–$0/1 \leftarrow$ **Verify**(pk, m, σ) is the verification algorithm. Given a public key pk, a signature σ and a message m, it outputs 1 if σ is a valid signature on m under pk and 0 otherwise.

For practical applications, this definition is required that the computation cost of the on-line re-signing phase is as small as possible. A divisible on-line/off-line proxy re-signature scheme is also a proxy re-signature scheme in which the re-signing algorithm is split into two sub-algorithms: the off-line re-signing algorithm and the on-line re-signing algorithm. A divisible on-line/off-line proxy re-signature scheme allows an adversary to query the signing/re-signing oracles with a message depending on this message's off-line re-signature token. By using divisible on-line/off-line proxy re-signature schemes, on-line/off-line threshold proxy re-signature schemes can be efficiently constructed through the simulation approach.

3 Simulation theorem for on-line/off-line threshold proxy re-signature schemes

3.1 On-line/off-line threshold proxy re-signature

We extend the definition of threshold proxy re-signature schemes [8,9] to define on-line/off-line threshold proxy re-signature schemes.

Definition 6 Assume that $\mathbb{P} = \{P_1, \cdots, P_n\}$ be a group of n proxies. An on-line/off-line threshold proxy re-signature scheme **OTPRS** is a tuple of algorithms (**KeyGen**, **T** – **ReKey**, **Sign**, **T** – **ReSign**$^{\textbf{off}}$, **T** – **ReSign**$^{\textbf{on}}$, **Verify**), where

–$(sk, pk) \leftarrow$ **KeyGen**(1^k) is the key generation algorithm. Given a security parameter $k \in \mathbb{N}$, this algorithm outputs signer's secret/public key pair (sk, pk).
–$(RSK_{A\to B}, RSK^1_{A\to B}, \cdots, RSK^n_{A\to B}) \leftarrow$ **T** – **Rekey**(sk_A, sk_B, pk_A, pk_B) is the threshold re-signature key generation algorithm. Given an (optional) delegatee's secret key sk_A, a delegator's secret key sk_B, and the corresponding public keys (pk_A, pk_B), it generates a re-signature key $RSK_{A\to B}$ and n re-signature key shares $RSK^i_{A\to B}$ of $RSK_{A\to B}$ where $RSK_{A\to B}$ is known to nobody and $RSK^i_{A\to B}$ is only known to proxy P_i.
–$\sigma \leftarrow$ **Sign**(sk, m) is the signing algorithm. Given a secret key sk and a message m, it generates a signature σ on m.
–$(\sigma_B^{off}, St_1, \cdots, St_n) \leftarrow$ **T** – **ReSign**$^{\textbf{off}}(pk_A, pk_B, RSK^1_{A\to B}, \cdots, RSK^n_{A\to B})$ is the off-line threshold re-signing algorithm. Given a delegatee's public key pk_A, a delegator's public key pk_B and the re-signature key shares $RSK^i_{A\to B}$ for $i = 1, \cdots, n$, it generates a public off-line re-signature token σ_B^{off} and a secret state information St_i for each proxy P_i.
–$\sigma_B^{on} \leftarrow$ **ReSign**$^{\textbf{on}}(RSK^1_{A\to B}, \cdots, RSK^n_{A\to B}, St_1, \cdots, St_n, pk_A, m, \sigma_A)$ is the on-line threshold re-signing algorithm. Given the re-signature key share $RSK^i_{A\to B}$ and the state information St_i of proxy P_i $(1 \le i \le n)$, a delegatee's public key pk_A, a message m and a signature σ_A, it generates an on-line re-signature token σ_B^{on} if **Verify**$(pk_A, m, \sigma_A) = 1$ and $\perp$ otherwise. The threshold re-signature for m is defined as $\sigma_B = (\sigma_B^{off}, \sigma_B^{on})$ which verifies under the delegator's public key pk_B.
–$0/1 \leftarrow$ **Verify**(pk, m, σ) is the verification algorithm. Given a public key pk, a signature σ and a message m, it outputs 1 if σ is a valid signature on m under pk and 0 otherwise.

3.2 The simulation theorem

Security notions of on-line/off-line threshold proxy re-signatures include unforgeability and robustness. Motivated by Gennaro *et al.*'s methodology [21] for proving the unforgeability of threshold signature schemes, we define the simulatability of **OTPRS** schemes. In this definition, the simulators can be seen as abstraction of real adversaries. The adversary is allowed to obtain the information of the corrupted proxies and the signature/re-signature of a message. This ensures that the executions of three protocols give the adversary no advantage in forging signatures for the **OTPRS** scheme. Namely, the adversary is allowed to get knowledge provided that such knowledge is worthless for forging signatures.

Definition 6 (*Simulatability*). Let **OTPRS**=(**KeyGen**, **T** – **ReKey**, **Sign**, **T** – **ReSign**$^{\textbf{off}}$, **T** – **ReSign**$^{\textbf{on}}$, **Verify**) be a (t, n) on-line/off-line threshold proxy re-signature

scheme. The scheme **OTPRS** is said to be simulatable if the following conditions hold.

1. The algorithm $\mathbf{T}-\mathbf{ReKey}$ is simulatable: There exists a simulator $SIM_{T-ReKey}$ that, on input a delegatee's public key pk_A and a delegator's public key pk_B, can simulate the view of the adversary on an execution of $\mathbf{T}-\mathbf{ReKey}$ which produces the public output (i.e., the verification key).
2. The algorithm $\mathbf{T}-\mathbf{ReSign^{off}}$ is simulatable: There exists a simulator $SIM_{T-ReSign^{off}}$ that, on input a delegatee's public key pk_A, a delegator's public key pk_B, t re-signature key shares $RSK^i_{A\to B}$ of the corrupted proxies and an off-line re-signature token σ_B^{off}, can simulate the view of the adversary on an execution of $\mathbf{T}-\mathbf{ReSign^{off}}$ which produces σ_B^{off} as an output.
3. The algorithm $\mathbf{T}-\mathbf{ReSign^{on}}$ is simulatable: There exists a simulator $SIM_{T-ReSign^{on}}$ that, on input a delegatee's public key pk_A, a delegator's public key pk_B, t re-signature key shares $RSK^i_{A\to B}$ of the corrupted proxies, a message m, an original signature σ_A and an on-line re-signature token σ_B^{on} on m, can simulate the view of the adversary on an execution of $\mathbf{T}-\mathbf{ReSign^{on}}$ which produces σ_B^{on} as an output.

We now prove the following theorem about the relationship between the security of an **OTPRS** scheme and the security of a **DOPRS** scheme.

Theorem 1 (simulation theorem). *An on-line/off-line threshold proxy re-signature scheme **OTPRS** is existentially unforgeable against adaptive chosen message attacks, if it is simulatable and its underlying divisible on-line/off-line proxy re-signature scheme **DOPRS** is existentially unforgeable against adaptive chosen message attacks.*

Proof. Let $\mathscr{A}$ denote an adversary of the on-line/off-line threshold proxy re-signature scheme **OTPRS** and $\mathscr{B}$ denote an adversary of the divisible on-line/off-line proxy re-signature scheme **DOPRS**. We show how $\mathscr{A}$ could help $\mathscr{B}$ to break the unforgeability of the underlying scheme **DOPRS** under the assumption that **OTPRS** is simulatable. First, we replace $\mathscr{A}$'s common parameter by $\mathscr{B}$'s common parameter. We then do the following.

When $\mathscr{A}$ issues a key generation query, we intercept and forward this query to $\mathscr{B}$'s challenger. The challenger returns the resulting key, and then we send it back to $\mathscr{A}$.

When $\mathscr{A}$ issues a re-signature key generation query on (pk_i, pk_j), we run the simulator $SIM_{T-ReKey}$ taking pk_i and pk_j as input, and return $SIM_{T-ReKey}$'s output to $\mathscr{A}$.

When $\mathscr{A}$ issues a signature query on (pk_i, m), we intercept it and forward (pk_i, m) as $\mathscr{B}$'s signature query to $\mathscr{B}$'s challenger. The challenger returns a corresponding signature σ. We then send σ back to $\mathscr{A}$.

When $\mathscr{A}$ issues an off-line re-signature query on (pk_i, pk_j), we intercept it and forward (pk_i, pk_j) as $\mathscr{B}$'s off-line re-signature query to $\mathscr{B}$'s challenger. The challenger returns a corresponding off-line re-signature token σ_j^{off}. We then run the simulator $SIM_{T-ReSign^{off}}$ taking pk_i, pk_j, σ_j^{off} and the transcript of the execution of $SIM_{T-ReKey}$ (which includes re-signature key shares of corrupted proxies) as input, and return $SIM_{T-ReSign^{off}}$'s output to $\mathscr{A}$.

When $\mathscr{A}$ issues an on-line re-signature query on $(pk_i, pk_j, m, \sigma_i)$, we intercept it and forward $(pk_i, pk_j, m, \sigma_i)$ as $\mathscr{B}$'s on-line re-signature query to $\mathscr{B}$'s challenger. The challenger returns a corresponding on-line re-signature token σ_j^{on}. We then run the simulator $SIM_{T-ReSign^{on}}$ taking pk_i, pk_j, m, σ_i, σ_j^{on} and the transcript of the executions of $SIM_{T-ReKey}$ and $SIM_{T-ReSign^{off}}$ as input, and return $SIM_{T-ReSign^{on}}$'s output to $\mathscr{A}$.

Finally, $\mathscr{A}$ outputs a pair (pk^*, m^*, σ^*), we intercept and return it as $\mathscr{B}$'s valid forgery. Since **OTPRS** is simulatable, $\mathscr{A}$'s view from the simulation is indistinguishable from its view in the real attack game.

Theorem 1 provides a way to construct on-line/off-line threshold proxy re-signature schemes: we can use a **DOPRS** scheme as a building block to construct an **OTPRS** scheme which is a threshold version of **DOPRS**. If **OTPRS** is simulatable according to Definition 6, the unforgeability of **OTPRS** can be reduced to that of its underlying scheme **DOPRS**.

Definition 9 (*Robustness*). A (t, n) on-line/off-line threshold proxy re-signature scheme **OTPRS** is said to be robust if it generates a correct output even in the presence of a malicious adversary that makes t corrupted proxies depart from the normal execution.

4 A divisible on-line/off-line proxy re-signature scheme

We assume that the message will be modified to a n_m-bit string before computing the signature or re-signature. We can achieve this by a collision-resistant hash function $L : \{0,1\}^* \to \{0,1\}^{n_m}$. The divisible on-line/off-line proxy re-signature scheme **DOPRS** = (**KeyGen**, **ReKey**, **Sign**, $\mathbf{ReSign^{off}}$, $\mathbf{ReSign^{on}}$, **Verify**), where

– **KeyGen**: Given a security parameter 1^k, choose two multiplicative cyclic groups G_1 and G_2 of the same prime order p. Assume that g is a generator of G_1, $H : \{0,1\}^* \to \mathbb{Z}_p^*$ is a collision-resistant hash function, and $e : G_1 \times G_1 \to G_2$ is an admissible pairing. Furthermore, choose $n_m + 2$ random elements $(g_2, u', u_1, \cdots, u_{n_m}) \in G_1^{n_m+2}$. Define a function $F(m) : \{0,1\}^{n_m} \to G_1$ mapping a n_m-bit string $Com = (Com_1, \cdots, Com_{n_m}) \in \{0,1\}^{n_m}$ onto $F(Com) = u' \prod_{i=1}^{n_m} (u_i)^{Com_i}$. The signer picks a random number $a \in \mathbb{Z}_p^*$, and computes the corresponding secret/public key pair $(sk, pk) = (a, g^a)$. The proxy chooses $y, z \in_R \mathbb{Z}_p^*$, and computes $Z = z^{-1} (\mathrm{mod}\, p)$, $h_1 = g^y$ and $h_2 = g^z$. The public parameter is

$cp = (G_1, G_2, H, p, g, F, e, u', h_1, h_2, u_1, \cdots, u_{n_m})$.

–**ReKey**: Given a delegatee's secret key $sk_A = a$ and a delegator's secret key $sk_B = b$, the proxy performs as follows:

1. Chooses and sends $w \in_R \mathbb{Z}_p^*$ to a delegatee. Then, the delegatee sends aw to a delegator. Furthermore, the delegator sends $b/(aw)$ to the proxy. Finally, the proxy computes $rk_{A\to B} = w \cdot b/(aw) = b/a$.
2. The secret re-signature key is set as $RSK = (rk_{A\to B}, y, z, Z)$.

–**Sign**: Given a secret key $sk = a$ and a massage $m' = (m'_1, ..., m'_{n_m}) \in \{0,1\}^{n_m}$, it generates a signature $\sigma = (\sigma_1, \sigma_2) = (g_2^a F(m')^r, g^r)$, where $r \in_R \mathbb{Z}_p^*$.

–**ReSign$^{\mathbf{off}}$**: Given a delegatee's public key $pk_A = g^a$, a delegator's public key $pk_B = g^b$ and a re-signature key $RSK_{A\to B} = (rk_{A\to B}, y, z, Z)$, the proxy performs as follows:

1. Chooses $m, r, s, r' \in_R \mathbb{Z}_p^*$ and computes $Com = g^m h_1^r h_2^s$.
2. Sends Com to the delegatee to obtain a signature $\sigma_A = (\sigma_{A,1}, \sigma_{A,2}) = (g_2^a F(Com)^{r_A}, g^{r_A})$ on Com, where r_A is randomly chosen from $\mathbb{Z}_p^*$ by the delegatee.
3. Checks that **Verify**$(pk_A, m, \sigma_A) \stackrel{?}{=} 1$. If σ_A is valid, the proxy chooses $r_B \in_R \mathbb{Z}_p^*$ and computes $\sigma_B^{off} = (pk_A, (\sigma_{A,1})^{rk_{A\to B}} F(Com)^{r_B}, (\sigma_{A,2})^{rk_{A\to B}} g^{r_B}) = (pk_A, g_2^b F(Com)^{r'_B}, g^{r'_B})$, where $r'_B = r_B + r_A \cdot b/a$. Otherwise, the proxy outputs $\perp$.
4. Computes $\theta = m + yr + sz \pmod p$.
5. Stores the state information $St = (\theta, m, r, s, r')$ and outputs the off-line re-signature token σ_B^{off}.

–**ReSign$^{\mathbf{on}}$**: On input a re-signature key $RSK_{A\to B} = (rk_{A\to B}, y, z, Z)$, a state information $St = (\theta, m, r, s, r')$, a delegatee's public key pk_A, a signature $\sigma'_A = (g_2^a F(m')^{r'_A}, g^{r'_A})$ and a message m', the proxy computes $s' = (\theta - H(m') - yr')Z \pmod p$ and outputs the on-line re-signature token $\sigma_B^{on} = (r', s', \sigma'_A)$. The re-signature for m' is defined as $\sigma'_B = (\sigma_B^{off}, \sigma_B^{on}) = (\sigma_{B,0}, \sigma_{B,1}, \sigma_{B,2}, \sigma_{B,3}, \sigma_{B,4}, \sigma_{B,5}, \sigma_{B,6}) = (pk_A, g_2^b F(Com)^{r'_B}, g^{r'_B}, r', s', g_2^a F(m)^{r'_A}, g^{r'_A})$. Note that $\sigma_B^{off} = (\sigma_{B,0}, \sigma_{B,1}, \sigma_{B,2}, \sigma_{B,3})$ might be sent to the recipient in the off-line stage.

–**Verify**: Given a public key pk, a purported signature σ and a message m', the verification algorithm performs as follows:

–If σ is an original signature $\sigma = (\sigma_1, \sigma_2)$, it checks that $e(\sigma_1, g) = e(\sigma_2, F(m'))e(pk, g_2)$. If the equation holds, it outputs 1; otherwise, outputs 0.

–If σ is a re-signature $\sigma = (\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6)$, it checks that $e(\sigma_1, g) = e(\sigma_2, F(g^{H(m')} h_1^{\sigma_3} h_2^{\sigma_4}))e(pk, g_2)$ $e(\sigma_5, g) = e(\sigma_6, F(m'))e(\sigma_0, g_2)$. If the above equations hold, it outputs 1; otherwise, it outputs 0.

Note that the **KeyGen** and **ReKey** algorithms are performed only once.

5 An efficient on-line/off-line threshold proxy re-signature scheme

Based on the divisible on-line/off-line proxy re-signature scheme given in Section 4, we construct an efficient on-line/off-line threshold proxy re-signature scheme.

5.1 Construction

Suppose that $\mathbb{P} = \{P_1, \cdots, P_n\}$ is a set of n proxies. Let **DOPRS**=(**KeyGen**, **ReKey**, **Sign**, **ReSign$^{\mathbf{off}}$**, **ReSign$^{\mathbf{on}}$**, **Verify**) be the divisible on-line/off-line proxy re-signature scheme described in Section 4. The resulting on-line/off-line threshold proxy re-signature scheme **OTPRS**=(**KeyGen**, **T − ReKey**, **Sign**,**T − ReSign$^{\mathbf{off}}$**, **T − ReSign$^{\mathbf{on}}$**, **Verify**) with a threshold $t < n/4$ is constructed as follows.

–(**KeyGen**, **Sign**, **Verify**) are the same as those in **DOPRS**.

–**T − ReKey**: The description of the threshold re-signature key generation algorithm is presented in Figure 1. This protocol is performed only once.

–**T − ReSign$^{\mathbf{off}}$**: The re-signing protocol for the off-line stage is presented in Figure 2.

–**T − ReSign$^{\mathbf{on}}$**: The re-signing protocol for the on-line stage is presented in Figure 3. Note that each proxy P_i first computes its re-signature share (r'_i, s'_i), then the proxies jointly combine all of the re-signature shares and generate an on-line re-signature token σ_B^{on}.

Robustness. The threshold re-signature key generation and the off-line threshold re-signing protocols can achieve robustness even if an adversary corrupted up to $n/3 - 1$ proxies. This is because, as observed in Section 2.2, all protocols (including DKG, INV, MUL and EC-Interpolate) are already robust against such kind of adversaries, or these protocols can easily be modified to achieve robustness. However, the proxies uses a polynomial of degree $2t$ to interpolate the value s'. If the adversary corrupts at most t proxies, then the number of proxies needed to correctly interpolate s' is $2t + 2t + 1$ according to the Berlekamp-Welch bound. It means that, even if at most t proxies behave dishonestly, $3t + 1$ honest proxies are still able to create a valid re-signature. Thus,

the whole protocol is robust against up to $t < n/4$ malicious proxies.

Threshold Re-Signature Key Generation Protocol

Public parameters : a security parameter k, a threshold t, two multiplicative cyclic groups G_1 and G_2 of the same prime order p, a bilinear pairing $e : G_1 \times G_1 \to G_2$, a generator g of G_1, $n_m + 2$ random elements $(g_2, u', u_1, \cdots, u_{n_m}) \in G_1^{n_m+2}$, a collision-resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_p^*$ and a function $F(m) : \{0,1\}^{n_m} \to G_1$ mapping a n_m-bit string $m = (m_1, \cdots, m_{n_m}) \in \{0,1\}^{n_m}$ onto $F(m) = u' \prod_{i=1}^{n_m} u_i^{m_i}$.
Private input : a delegatee's key pair $(pk_A, sk_A) = (g^a, a)$ and a delegator's key pair $(pk_B, sk_B) = (g^b, b)$.
Public output : the public parameters $(h_1, h_2, \{vk_i\}_{i=1}^n)$.
Private output (**for eachproxy $\mathbf{P_i}$**) : a share $RSK^i_{A\to B}$ of the re-signature key $RSK_{A\to B}$.

1. The proxies jointly run the DKG protocol twice in order to get two public values h_1 and h_2. We use y_i and z_i to denote the shares of two secret values y, z (such that $h_1 = g^y$ and $h_2 = g^z$) held by proxy P_i.
2. The proxies jointly run the INV protocol so that proxy P_i obtains a share Z_i of the inverse Z of z.
3. Each proxy P_i first sends $w_i \in_R \mathbb{Z}_p^*$ to the delegatee. Then, the delegatee sends aw_i to the delegator. Next, the delegator chooses a random polynomial $f(x) \in \mathbb{Z}_p[x]$ of degree t such that $f(0) = b$. Furthermore, the delegator computes $b_i = f(i)$ and $vk_i = g^{b_i}$, and sends $(b_i/(aw_i), vk_i)$ to proxy P_i. Finally, P_i computes $rk^i_{A\to B} = w_i \cdot b_i/(aw_i) = b_i/a$. We denote with vk_i the verification key of proxy P_i.
4. The public parameters are

$$params = (n, t, p, g, g_2, F, u', \{u_i\}_{i=1}^{n_m}, \{vk_i\}_{i=1}^n),$$

while each proxy P_i secretly retains $RSK_i = (rk^i_{A\to B}, y_i, z_i, Z_i)$ as his own local re-signing key share.

Figure 1. **T – ReKey**

The off-line threshold re-signing algorithm is performed per message. This algorithm is described as follows.

Off-Line Threshold Re-Signing Protocol

Public input : a delegatee's public key $pk_A = g^a$ and a delegator's public key $pk_B = g^b$.
Private input (**for proxy $\mathbf{P_i}$**) : the re-signature key share $RSK_i = (rk^i_{A\to B}, y_i, z_i, Z_i)$.
Public output : an off-line re-signature token σ_B^{off}.
Private output (**for proxy $\mathbf{P_i}$**) : a secret state information $St_i = (\theta_i, m_i, r_i, s_i, r_i', \mu_i)$.

1. The proxies jointly run the DKG protocol four times to generate four shared random values m, r, s, r'. We denote with m_i, r_i, s_i and r_i' the shares of the secret values m, r, s, r' (such that $h_3 = g^m$, $h_4 = h_1^r$, $h_5 = h_2^s$, $h_6 = h_1^{r'}$) held by proxy P_i.
2. The proxies jointly run the DKG protocol to generate μ_i for each proxy P_i through a degree $2t$ polynomial $f_0(x) \in \mathbb{Z}_p[x]$ such that $f_0(0) = 0$.
3. Now, h_3, h_4 and h_5 are known to the proxies, so $Com = h_3h_4h_5 = g^m h_1^r h_2^s$.
4. The delegatee generates a signature $\sigma_A = (\sigma_{A,1}, \sigma_{A,2}) = (g_2^a F(Com)^{r_A}, g^{r_A})$ on Com, where r_A is randomly chosen from $\mathbb{Z}_p^*$.
5. Proxy P_i first verifies **Verify**$(pk_A, Com, \sigma_A) \stackrel{?}{=} 1$. If it does not hold, outputs $\perp$; otherwise, P_i chooses $r_{B,i} \in_R \mathbb{Z}_p^*$ and broadcasts its re-signature share

$$\begin{aligned}\sigma_{B_i} &= (\sigma_{B_i,1}, \sigma_{B_i,2}) \\ &= ((\sigma_{A,1})^{rk^i_{A\to B}} F(Com)^{r_{B,i}}, (\sigma_{A,2})^{rk^i_{A\to B}} g^{r_{B,i}}) \\ &= (g_2^{ab_i/a} F(Com)^{r_{B,i}+r_A b_i/a}, g^{r_{B,i}+r_A b_i/a}) \\ &= (g_2^{b_i} F(Com)^{r'_{B,i}}, g^{r'_{B,i}}),\end{aligned}$$

where $r'_{B,i} = r_{B,i} + r_A b_i/a$. Each proxy P_j verifies the share $\sigma_{B_i} = (\sigma_{B_i,1}, \sigma_{B_i,2})$ from the other proxies. P_j verifies $e(\sigma_{B_i,1}, g) \stackrel{?}{=} e(F(Com), \sigma_{B_i,2}) e(vk_i, pk_A)$. If the equation does not verify, P_j broadcasts a complaint against P_i. When the total number of complaints is more than t, P_i is regarded as disqualification. Let $QUAL$ be the set of

qualified proxies. If $|QUAL| > t$, the proxies can compute the off-line re-signature token

$$\begin{aligned}\sigma_B^{off} &= (\sigma_{B,0}, \sigma_{B,1}, \sigma_{B,2}) \\ &= (pk_A, \prod_{i\in QUAL} \sigma_{B_i,1}^{\lambda_{i,0}}, \prod_{i\in QUAL} \sigma_{B_i,2}^{\lambda_{i,0}}) \\ &= (pk_A, \prod_{i\in QUAL} g_2^{b_i\lambda_{i,0}} F(Com)^{r'_{B,i}\lambda_{i,0}}, \prod_{i\in QUAL} g^{r'_{B,i}\lambda_{i,0}}) \\ &= (pk_A, g_2^b F(Com)^{r'_B}, g^{r'_B}),\end{aligned}$$

where $r'_B = \sum_{i\in QUAL} r'_{B,i}\lambda_{i,0}$ and $\lambda_{i,0}$ is the Lagrange interpolation coefficient.

6. The proxies jointly run the MUL protocol twice to compute the shares of the products ry and sz. Then, the proxies (non interactively) compute shares of the value $\theta = m + ry + sz \pmod p$. We use θ_i to denote the share of θ held by proxy P_i.
7. The off-line re-signature token is $\sigma_B^{off} = (pk_A, g_2^b F(Com)^{r'_B}, g^{r'_B})$, while the state information for proxy P_i is $St_i = (\theta_i, m_i, r_i, s_i, r', \mu_i)$.

Figure 2. **T – ReSign^{off}**

Note that the efficiency of the on-line threshold re-signing algorithm is the most important when optimizing threshold re-signature computation. The off-line phase of our scheme requires several rounds of communication, but the on-line stage is typically efficient. The proxies can compute a re-signature in a short time once the message to be re-signed is presented.

On-Line Threshold Re-Signing Protocol

Public intput : a delegatee's public key pk_A, a message $m' \in \mathbb{Z}_q$ and a signature σ'_A.
Private input (**for proxy $\mathbf{P_i}$**) : the re-signature key share RSK_i $=(rk^i_{A\to B}, y_i, z_i, Z_i)$ and the state information $St_i = (\theta_i, m_i, r_i, s_i, r', \mu_i)$.
Public output : an on-line re-signature token σ_B^{on}.

1. Each proxy P_i locally computes the share $s'_i = (\theta_i - H(m')r'y)Z_i + \mu_i \pmod p$. Then, the proxies broadcasts their shares s'_i to compute $s' =$EC-Interplate$(s'_1, \cdots, s'_n)$.
2. The on-line re-signature token for m' is given by (r', s', σ'_A).

Figure 3. **T – ReSign^{on}**

5.2 Security

Theorem 2. *The proposed on-line/off-line threshold proxy re-signature scheme **OTPRS** is simulatable.*

Proof. The simulation techniques used here are similar to those used in [21,22]. We give simulators for **T – ReKey**, **T – ReSign^{off}**, **T – ReSign^{on}**, respectively. Without loss of generality, suppose that the proxies $P_1, \cdots, P_t$ are corrupted by an adversary.

Simulator $SIM_{T-ReKey}$ for the protocol **T – ReKey**: Given two public keys $(pk_A, pk_B) = (g^a, g^b)$, $SIM_{T-ReKey}$ performs as follows.

1. The execution of the DKG protocol is replaced by the execution of the simulator SIM_{DKG} for the DKG protocol. $SIM_{T-ReKey}$ runs $SIM_{DKG}(h_1)$ to fix the result of the DKG execution to be h_1. $SIM_{T-ReKey}$ runs $SIM_{DKG}(h_2)$ to fix the result of the DKG execution to be h_2. At the end, proxy $P_i (1 \le i \le n)$ obtains its shares (y_i, z_i).
2. The INV protocol is run exactly as in the real protocol.
3. Step 3 is done as follows. The simulator $SIM_{T-ReKey}$ first chooses t random numbers c_i from $\mathbb{Z}_p^*$, and computes $vk_i = g^{c_i}$ for $i = 1, \cdots, t$. Using Lagrange interpolation, $SIM_{T-ReKey}$ then constructs $g^{f(x)}$ such that $g^{f(0)} = g^b = pk_B$ and $g^{f(i)} = g^{c_i}$ for $i = 1, \cdots, t$. Finally, $SIM_{T-ReKey}$ broadcasts $vk_j = g^{f(j)}$ $(j = t+1, \cdots, n)$ for the uncorrupted proxies.
4. $SIM_{T-ReKey}$ executes the rest of this protocol as in the real protocol.

Note that $SIM_{T-ReKey}$ provides the adversary with a view (public outputs and private outputs of corrupted proxies) which is exactly indistinguishable from actual execution.

Simulator $SIM_{T-ReSign^{off}}$ for the protocol **T – ReSign^{off}**: Given two public keys $(pk_A, pk_B) = (g^a, g^b)$, t re-signature key shares $RSK^i_{A\to B}$ and an off-line re-signature token $\sigma_B^{off} = (\sigma_{B,1}, \sigma_{B,2})$, $SIM_{T-ReSign^{off}}$ performs as follows.

1. $SIM_{T-ReSign^{off}}$ executes $SIM_{DKG}(g^m)$, $SIM_{DKG}(h_1^r)$, $SIM_{DKG}(h_2^s)$ and $SIM_{DKG}(h_1^{r'})$. As a result, proxy $P_i (1 \le i \le n)$ obtains its shares (m_i, r_i, s_i, r'_i).

2. $SIM_{T-ReSign^{off}}$ runs SIM_{DKG} to simulate an execution of the DKG protocol such that the output is fixed to 0. Proxy $P_i (1 \leq i \leq n)$ obtains its share μ_i.
3. Steps 3, 4, 6 and 7 are run exactly as in the real protocol, so we concentrate on step 5. Let $Com = g^m h_1^r h_2^s$ and $\theta = m + ry + sz \pmod p$. $SIM_{T-ReSign^{off}}$ first chooses $r_{B,i} \in_R \mathbb{Z}_p^*$ and computes the share $\sigma_{B_i} = (\sigma_{B_i,1}, \sigma_{B_i,2}) = ((\sigma_{A,1})^{rk^i_{A\to B}} F(Com)^{r_{B,i}}, (\sigma_{A,2})^{rk^i_{A\to B}} g^{r_{B,i}})$ by the re-signature key share $RSK_i = (rk^i_{A\to B}, y_i, z_i, Z_i)$ for $i = 1, \cdots, t$, where $\sigma_A = (\sigma_{A,1}, \sigma_{A,2}) = ((pk_A)^{-J(Com)/K(Com)} F(Com)^{r_A}, (pk_A)^{-1/K(Com)} g^{r_A})$. For the uncorrupted proxies, $SIM_{T-ReSign^{off}}$ uses Lagrange interpolation to compute the shares $\sigma_{B_j} = (\sigma_{B_j,0}, \sigma_{B_j,1}, \sigma_{B_j,2}) = (pk_A, \sigma_{B,1}^{\lambda_{0,j}} \prod_{l=1}^{t} \sigma_{B_i,1}^{\lambda_{l,j}}, \sigma_{B,2}^{\lambda_{0,j}} \prod_{l=1}^{t} \sigma_{B_i,2}^{\lambda_{l,j}})$ for $j = t+1, \cdots, n$.

The above simulation is perfectly indistinguishable from a real execution of the protocol.

Simulator $SIM_{T-ReSign^{on}}$ for the protocol **T − ReSign^{on}**: Given two public keys $(pk_A, pk_B) = (g^a, g^b)$, t re-signature key shares $RSK^i_{A\to B}$ of the corrupted proxies, a message m', an original signature σ'_A and an on-line re-signature token $\sigma^{on}_B = (r', s', \sigma'_A)$, $SIM_{T-ReSign^{on}}$ works as follows.

1. $SIM_{T-ReSign^{on}}$ computes the share $s'_i = (\theta_i - H(m') - y_i r')Z_i + \mu_i \pmod p$ for $i = 1, \cdots, t$. $SIM_{T-ReSign^{on}}$ then chooses a $2t$-degree polynomial $f_s(x) \in \mathbb{Z}_p[x]$ such that $f_s(0) = s'$ and $f_s(i) = s'_i$ for $i = 1, \cdots, t$. For the uncorrupted proxies, $SIM_{T-ReSign^{on}}$ broadcasts $s'_j = f_s(j)$ for $j = t+1, \cdots, n$. Notice that all proxies can compute s' =EC-Interplate$(s'_1, \cdots, s'_n)$. The interpolation of these values s'_j together with shares s'_j of the corrupted proxies comes out to be s', just as in the real protocol.

Since the simulator does not know the secret signing key and the re-signature key, it is clear that the adversary's view is exactly in the same as in the actual protocol.

From Theorems 1 and 2, we can derive the following theorem.

Theorem 3. *If the CDH problem and the discrete logarithm problem are intractable, the proposed* ***OTPRS*** *scheme is existentially unforgeable and robust against an adversary who corrupts up to $t < n/4$ malicious proxies.*

5.3 Comparison

We compare the proposed scheme with some existing threshold proxy re-signature schemes in Table 1. We mainly analyze bit complexity of all schemes required by a proxy in computing its re-signature share when a message to be re-signed arrives. Note that all proxies compute their re-signature shares in parallel. However, schemes **BTPRS11** [8], **UTPRS11** [8] and **BTPRS10** [9] are not considered to be on-line/off-line threshold proxy re-signature schemes because no pre-computation is performed. To achieve the same level of security, assume that the modular parameter p in all schemes is the same size.

Table 1: Comparison of computational complexity among threshold proxy re-signature schemes

Schemes	Additions	Subtractions	Multiplications	Exponentiations	Pairings
BTPRS11	0	0	0	2	3
UTPRS11	0	0	1	3	2
BTPRS10	0	0	2	4	3
Our **OTPRS**	1	2	2	0	0

Referring to Table 1, in our **OTPRS** scheme, each proxy performs one modular addition, two modular subtractions and two modular multiplications when computing an re-signature share. This is very efficient and comparable to the other schemes. Our scheme makes a tradeoff by incurring a large cost in the off-line stage to obtain a quick on-line stage.

6 Conclusion

We give a simulation theorem for on-line/off-line threshold proxy re-signature schemes. We also propose a divisible on-line/off-line proxy re-signature scheme. Fruthermore, we propose an efficient on-line/off-line threshold proxy re-signature scheme based on the simulation theorem and the proposed divisible scheme. Compared with existing threshold proxy re-signature schemes, the advantages of our new scheme is lower costs of computation when the messages to be re-signed arrive. The security of our **OTPRS** scheme can be reduced to that of its underlying **DOPRS** scheme together with the discrete logarithm assumption.

Acknowledgement

This research is supported in part by the National Natural Science Foundation of China (Grant Nos. 61262057 and 61202362), Natural Science Foundation of Gansu Province of China (Grant No. 145RJDA325 and 1308RJYA039), Science and Technology Project of State Archives Administration of China (Grant No. 2014-X-33), Science and Technology Project of Lanzhou City of China (Grant No. 2013-4-22) and Youth Foundation of Northwest Normal University (Grant No. NWNU-LKQN-12-23). The authors are grateful to the anonymous referee for a careful checking of the details and for helpful comments that improved this paper.

References

[1] M. Blaze, G. Bleumer, and M. Strauss, Divertible protocols and atomic proxy cryptography, Proceedings of EUROCRYPT 1998, pp. 127-144 (1998).

[2] X.D. Yang, C.F. Wang, L. Zhang, et al., On-line/off-line threshold proxy re-signatures, Chinese Journal of Electronics, 23, 248-253 (2014).

[3] Y.Q. Deng and G. Song, Proxy re-signature scheme based on quadratic residues, Journal of Networks, 10, 1459-1465 (2011).

[4] J. Shao, G. Wei, Y. Ling, M. Xie, Unidirectional identity-based proxy re-signature, Proceedings of 2011 IEEE ICC, Kyoto, Japan, pp. 1-5 (2012).

[5] D.F. He, A novel blind proxy re-signature scheme, Computer Applications and Software, 29, 294-296 (2012).

[6] H. Shao, F. Zhang, X.Z. Yuan, et al., A proxy signature based on the difficulty of solving equations of higher on quaternion ring, Applied Mechanics and Materials, 51, 1951-1954 (2014).

[7] J. Shao, Z.F. Cao, L. Wang, and X. Liang, Proxy re-signature schemes without random oracles, Proceedings of INDOCRYPT 2007, Chennai, India, pp. 197-209 (2007).

[8] P.Y. Yang, Z.F. Cao, and X. Dong, Threshold proxy re-signature, Journal of Systems Science and Complexity, 24, 816-824 (2011).

[9] X.D. Yang and C.F. Wang, Threshold proxy relsignature schemes in the standard model, Chinese Journal of Electronics, 20, 691-696 (2011).

[10] E. Bresson, D. Catalano, and R. Gennaro, Improved on-line/off-line threshold signatures, Proceedings of PKC 2007, pp. 217-232 (2007).

[11] C. Crutchfield, D. Molnar, D. Turner, and D. Wagner, Generic on-line/off-line threshold signatures, Proceedings of PKC 2006, pp. 58-74 (2006).

[12] C. Gao, B. Wei, D. Xie, and C. Tang, Divisible on-line/off-line signatures, Proceedings of CT-RSA 2009, pp. 562-566 (2009).

[13] H. Krawczyk and T. Rabin, Chameleon hashing and signatures, in Proceedings of the 7th Annual Network and Distributed System Security Symposium, pp. 143-154 (2000).

[14] A. Shamir and Y. Tauman, Improved on-line/off-line signature schemes, Proceedings of CRYPTO'01, LNCS 2139, pp. 355-367 (2001).

[15] A. Shamir, How to share a secret, Communications of the ACM, 22, 612-613, (1979).

[16] Welch L, Berlekamp E. Error correction of algebraic block codes. US Patent Number 4633470, issued December 1986.

[17] M. Ben-Or, S. Goldwasser, A. Wigderson, Completeness theorems for non-cryptographic fault tolerant distributed computation, Proceedings of the 20th Annual ACM on STO, New York, USA, pp. 1-10, (1988).

[18] J. Bar-Ilan,D. Beaver, Non-cryptographic fault-tolerant computing in constant number of rounds of interaction, Proceedings of the eighth annual ACM Symposium on PPODC, New York, USA, pp. 201-209, (1989).

[19] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, Secure distributed key generation for discrete-log based cryptosystems, Proceedings of EUROCRYPT'99, Springer: Berlin, pp. 295-310 (1999).

[20] J. Shao, M. Feng, B. Zhu, Z.F. Cao, and P. Liu, The security model of unidirectional proxy re-signature with private re-signature key, Information Systems Security, LNCS 6168, pp. 216-232 (2010).

[21] R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, Adaptive security for threshold cryptosystems, Proceedings of CRYPTO 1999, LNCS 1666, pp. 98-115, (1999).

[22] V. Shoup, R. Gennaro, Securing threshold cryptosystems against chosen ciphertext attack. Journal of Cryptology, 15, 75-96, (2002).

Xiaodong YANG is an associate professor of Information and Computer Science at Northwest Normal University. He received his M.S. degree in mathematics from Tongji University in 2005 and Ph.D. degree in cryptography from Northwest Normal University in 2010. His main research interests include information security and computer cryptography.

Guojuan Gao is now a postgraduate student of cryptography at Northwest Normal University. Her current research interests include network security and cryptography.

Yanan Li is now a postgraduate student of cryptography at Northwest Normal University. His current research interests include information security and cloud security.

Yan Li is now a postgraduate student of cryptography at Northwest Normal University. Her current research interests include proxy signature and proxy re-signature.

Caifen Wang is a professor of Information and Computer Science at Northwest Normal University. She received her M.S. degree in compute science from Lanzhou University in 1998 and Ph.D. degree in cryptography from Xidian University in 2003. Her current research interests include network security, cryptographic protocols and electronic commerce. She is also a member of Chinese Cryptology and Information Security Association.