Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/090224

Divisible On-Line/Off-Line Proxy Re-Signature

Xiaodong Yang[∗], *Chunmei Li, Yan Li, Sian Zhou and Caifen Wang*

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

Received: 26 May 2014, Revised: 25 Aug. 2014, Accepted: 27 Aug. 2014
Published online: 1 Mar. 2015

Abstract: To improve the real-time efficiency of proxy re-signature schemes, a notion called divisible on-line/off-line proxy re-signatures is introduced in this paper. The idea of this notion is to split the re-signing algorithm into two phases: the off-line and on-line phases. The proxy can perform the bulk of re-signature computation in the off-line phase before seeing the message to be re-signed. The results of this pre-computation are saved and then used in the on-line phase after the message to be re-signed is given. Divisible on-line/off-line proxy re-signatures are very useful in a particular scenario where the proxy must respond quickly once the message to be re-signed is presented. In a divisible on-line/off-line proxy re-signature scheme, the partial re-signature might be exposed before the message to be re-signed arrives. We then propose a generic divisible on-line/off-line proxy re-signature scheme, which can transform any proxy re-signature scheme into a highly efficient divisible on-line/off-line proxy re-signature scheme. Compared to existing proxy re-signature schemes, our scheme demonstrates higher efficiency. The security of the scheme relies on the unforgeability of its underlying proxy re-signature scheme and the difficulty to solve the discrete logarithm problem.

Keywords: divisible on-line/off-line proxy re-signature, on-line/off-line proxy re-signature, unforgeability

1 Introduction

Proxy re-signature, introduced by Blaze, Bleumer and Strauss [1], allows a semi-trusted proxy to convert a delegatee's signature into a delegator's signature on the same message by using some additional information (a.k.a., re-signature key). However, the proxy does not obtain any signing key and cannot sign arbitrary messages on behalf of the delegatee or the delegator on its own. In a bidirectional scheme, the re-signature key allows the proxy to transform a delegatee's signature into a delegator's signature as well as a delegator's signature into a delegatee's signature. In a unidirectional scheme, the re-signature key allows the proxy to transform a delegatee's signature into a delegator's signature, but not vice verse. Due to the transformation function, proxy re-signature schemes are very useful and can be applied in many applications, including simplifying key management, forming weak group signatures, and constructing digital rights management (DRM) interoperable system, etc. Many constructions of proxy re-signature schemes appear in the literature [2,3,4,5,6,7,8,9,10], but most of these schemes are not fast enough for many practical applications. Proxy re-signature schemes that are efficient and provably secure are interesting both from a practical and a theoretical point of view.

In this paper, we introduce a notion called divisible on-line/off-line proxy re-signatures to improve the performance of proxy re-signature schemes, and we show how to construct such proxy re-signatures from existing proxy re-signature schemes. The idea of this notion is to divide the re-signature generation algorithm into two phases. The first phase is performed off-line (without knowing the message to be re-signed) and the second phase is performed on-line (after knowing the message to be re-signed). In the off-line phase, the proxy handle the most costly computations. Once the message to be re-signed is presented, the proxy utilizes the results of the pre-computation and produces a corresponding re-signature in a very short time. At that point, the computation of the actual re-signature requires very little effort. When re-signing a message m', a partial re-signature of m' called an off-line re-signature token is first computed in the off-line phase. The remaining part of the re-signature of m', called the on-line re-signature token, is generated in the on-line phase when the message is known. The off-line re-signature token is allowed to be exposed in the off-line phase. In other word, a divisible on-line/off-line proxy re-signature scheme is still secure if the adversary is allowed to query the signing/re-signing

[∗] Corresponding author e-mail: y200888@163.com oracle with a message depending on this message's off-line re-signature token.

Motivations: There are two reasons to consider divisible on-line/off-line proxy re-signatures:

1. Divisible on-line/off-line proxy re-signatures could be suitable for a scenario where the proxy must respond quickly once the message to be re-signed is presented. For example, divisible on-line/off-line proxy re-signature schemes are particularly useful in smart card applications: The off-line phase is performed either during the card manufacturing process or as a background computation whenever the card is connected to power, and the on-line phase utilizes the stored results of the off-line phase to re-sign actual messages. The on-line phase is typically very fast, and hence can be executed efficiently even on a weak processor.
2. The divisible on-line/off-line proxy re-signature scheme can send the off-line re-signature token in the off-line phase instead of in the on-line phase. This reduces the on-line bandwidth of the communication channel. For example, the proxy can pre-compute a series of off-line re-signature tokens and transmits these tokens when the communication channel is not busy.

Related work: The primitive of proxy re-signature was introduced at Eurocrypt'98 by Blaze, Bleaumer and Strauss [1]. They proposed the first proxy re-signature scheme that is bidirectional and multi-use. However, from the re-signature key (which is public), the delegatee can easily get the delegator's signing key or vice versa. In 2005, Ateniese and Hohenberger [2] first formalized the definition of security for proxy re-signature, and then presented two proxy re-signature schemes. The first scheme S_{bi} is bidirectional and multi-use, and the second scheme S_{uni} is unidirectional and single-use. The security of all schemes are analyzed in the random oracle model. However, Gennaro *et al.* [11] point out that some schemes are proven secure in the random oracle model, but they are trivially insecure under any instantiation of the oracle. Later, Shao *et al.* [12] proposed a bidirectional and multi-use proxy re-signature scheme S_{mb} without random oracle, and Chow and Phan [13] proposed a unidirectional and single-use proxy re-signature scheme *CP-PRS* in the standard model. Libert and Vergnaud [14] proposed a unidirectional and multi-use proxy re-signature scheme *LV-PRS* in the standard model. To our knowledge, there is no divisible on-line/off-line proxy re-signature scheme existed in the literature yet.

The basic idea of our scheme makes use of chameleon hashing to construct divisible on-line/off-line proxy re-signatures. A chameleon hash function is a special hash function endowed with a public hash key and a secret trapdoor key. The function is collision-resistant unless one knows the secret trapdoor key. Knowledge of the public hash key allows one to compute the hash function, while knowledge of the secret trapdoor key allows one to find arbitrary collisions. For many chameleon hash functions [15,16,17,18,19], the collision-finding procedure is very efficient, requiring only several modular multiplications.

Our contribution: We first explicitly give and exemplify the notion of divisible on-line/off-line proxy re-signatures. We show how to combine any proxy re-signature scheme with a specific chameleon hash function based on the discrete logarithm problem, and then present a generic divisible on-line/off-line proxy re-signature scheme, which can convert any proxy re-signature scheme into a divisible on-line/off-line proxy re-signature scheme. Our scheme is proven secure without resorting to the random oracle model. When compared to the existing proxy re-signature schemes, the proposed scheme requires less computation cost. The on-line complexity of our scheme is equivalent to two modular subtraction and two modular multiplication.

Organization: Our paper is organized as follows. We review some definitions in Section 2. The security model of the divisible on-line/off-line proxy re-signature scheme is presented in Section 3. The proposed scheme is given in Section 4. Conclusion is offered in Section 5.

2 Preliminaries

In this section, we review some background knowledge used in this paper, including negligible function, the discrete logarithm assumption and the formal definition of the proxy re-signature scheme.

Definition 1 (Negligible function). A function $\psi : \mathbb{N} \to \mathbb{R}$ is negligible if for all $c > 0,\ \psi(k) < 1/k^c$ for all sufficiently large k.

Definition 2 (Discrete logarithm assumption). Given a group G of prime order p with a generator g and element g^x where x is chosen at random from $\mathbb{Z}_p^*$, the discrete logarithm problem in G is to compute x. We say that the (ε, t) discrete logarithm assumption holds in a group G if no algorithm running in time at most t can solve the discrete logarithm problem in G with probability at least ε.

Definition 3 (Proxy re-signature scheme). A proxy re-signature scheme **PRS** consists of algorithms **KeyGen**, **ReKey**, **Sign**, **ReSign** and **Verify**.

– (**KeyGen**, **Sign**, **Verify**) form the key generation, signing and verification algorithms of a standard signature scheme.

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

–$\sigma_B \leftarrow$ **ReSign**$(rk_{A\to B}, pk_A, m, \sigma_A)$ is the re-signing algorithm. On input public parameters cp, a re-signature key $rk_{A\to B}$, a delegatee's public key pk_A, a message m and a signature σ_A, this algorithm outputs a re-signature σ_B if **Verify**(pk_A, m, σ_A)=1 and outputs $\perp$ otherwise.

3 Security model and security notions

We extend the definition of a proxy re-signature scheme to define a divisible on-line/off-line proxy re-signature scheme.

Definition 4 (Divisible on-line/off-line proxy re-signature scheme). A divisible on-line/off-line proxy re-signature scheme **DOPRS** = (**KeyGen**, **ReKey**$^{\mathbf{on/off}}$, **Sign**, **ReSign**$^{\mathbf{off}}$, **ReSign**$^{\mathbf{on}}$, **Ver**) consists of the following algorithms:

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

–$RSK_{A\to B} \leftarrow$ **ReKey**$^{\mathbf{on/off}}(sk_A, sk_B, pk_A, pk_B)$ is the re-signature key generation algorithm. On input global parameters $params$, an (optional) delegatee's private key sk_A, a delegator's private 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 delegatee's signatures into delegator's signatures.

–$\sigma \leftarrow$ **Sign**(sk, m) is the signing algorithm. On input global parameters $params$, a secret key sk and a message m, this algorithm outputs a signature σ on the message m.

–$(\sigma_B^{off}, St) \leftarrow$ **ReSign**$^{\mathbf{off}}(RSK_{A\to B}, pk_A, pk_B)$ is the off-line re-signing algorithm. On input global parameters $params$, a delegatee's public key pk_A, a delegator's public key pk_B and a re-signature key $RSK_{A\to B}$, this algorithm outputs a (public) off-line re-signature token σ_B^{off} and a (secret) state information St. The state information is kept secret and will be transmitted to the execution of the on-line re-singing algorithm.

–$\sigma_B^{on} \leftarrow$ **ReSign**$^{\mathbf{on}}(RSK_{A\to B}, St, pk_A, m, \sigma_A)$ is the on-line re-signing algorithm. On input global parameters $params$, 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, this algorithm first checks that σ_A is valid w.r.t. pk_A. It outputs an on-line re-signature token σ_B^{on} if **Ver**$(pk_A, m, \sigma_A) = 1$ and $\perp$ otherwise. The re-signature for m is defined as $\sigma_B = (\sigma_B^{off}, \sigma_B^{on})$ which is verified under a delegator's public key pk_B.

–$0/1 \leftarrow$ **Ver**(pk, m, σ) is the verification algorithm. On input global parameters $params$, a public key pk, a message m and a purported signature σ, this algorithm outputs 1 if σ is a valid signature under pk and 0 otherwise.

Completeness: For any message m in the message space, the following two conditions must hold:

$$\mathbf{Ver}(pk_A, m, \sigma_A) = 1 \text{ and } \mathbf{Ver}(pk_B, m, \sigma_B) = 1,$$

where $\sigma_B = (\sigma_B^{off}, \sigma_B^{on})$.

Remark 1: In a divisible on-line/off-line proxy re-signature scheme, a signature manifests in two types: the original signature and the re-signature. An original signature can be computed only by the owner of the signing key, while a re-signature can be computed not only by the owner of the signing key, but also by collaboration between his proxy and delegatee. Namely, an original signature is one which is only outputted by the algorithm **Sign**.

Remark 2: ReSign$^{\mathbf{off}}$ and **ReSign**$^{\mathbf{on}}$ can be viewed as sub-algorithms of a complete re-signing algorithm. For simplicity, we use the notation $(\sigma_B^{off}, \sigma_B^{on}) \leftarrow (\mathbf{ReSign^{off}}, \mathbf{ReSign^{on}})(RSK_{A\to B}, pk_A, m, \sigma_A)$ to denote such a complete re-signing process: $(\sigma_B^{off}, St) \leftarrow \mathbf{ReSign^{off}}(RSK_{A\to B}, pk_A, pk_B)$ and $\sigma_B^{on} \leftarrow \mathbf{ReSign^{on}}(RSK_{A\to B}, St, pk_A, m, \sigma_A)$.

In the following, we define a security model for divisible on-line/off-line proxy re-signature schemes, which is an extension of the security model for proxy re-signature schemes presented by Ateniese and Hohenberger [2], and Shao *et al.* [20]. The major difference between two models is that the attacker in the new model is allowed to adaptively select the query messages depended on their off-line re-signature tokens. In the security model of a divisible on-line/off-line proxy re-signature scheme, the adversary has obtained the off-line re-signature token of a message before he queries this message. The adversary $\mathscr{A}$ is allowed to make queries to an off-line re-signing oracle $\mathscr{O}_{ReSign^{off}}$ and an on-line re-signing oracle $\mathscr{O}_{ReSign^{on}}$. We assume that if $\mathscr{A}$ makes the i-th on-line re-signature query, then $\mathscr{A}$ has already made the i-th off-line re-signature query. The security model for divisible on-line/off-line proxy re-signature schemes contains two aspects: the external security and the internal security. The details are as follows.

External Security: This security notion deals with adversaries outside the system (i.e., neither the proxy nor the delegation parties). A divisible on-line/off-line proxy re-signature scheme is said to be external-secure if for the security parameter k, any non-zero n, and all probabilistic polynomial-time algorithms $\mathscr{A}$, the following probability is negligible:

$$\begin{aligned}&\Pr[\{(pk_i, sk_i) \leftarrow \mathbf{KeyGen}(1^k)\}_{i\in\{1,\dots,n\}},\\ &(i^*, m^*, \sigma^*) \leftarrow \mathscr{A}^{\mathscr{O}_{Sign}(\cdot,\cdot),\mathscr{O}_{ReSign^{off}}(\cdot,\cdot),\mathscr{O}_{ReSign^{on}}(\cdot,\cdot,\cdot,\cdot)}(\{pk_i\}_{i\in\{1,\dots,n\}}):\\ &\mathbf{Ver}(pk_{i^*}, m^*, \sigma^*) = 1 \wedge (i^*, m^*) \notin \mathscr{Q}]\end{aligned}$$

where $\mathscr{O}_{Sign}(\cdot,\cdot)$ is a signing oracle taking as input a message m and an index $i \in \{1, \dots, n\}$ to return the output of $\sigma \leftarrow$**Sign**(sk_i, m); the off-line re-signing oracle $\mathcal{O}_{ReSign^{off}}(\cdot,\cdot)$ takes as input two distinct indices $i, j \in \{1, \ldots, n\}$, and returns the output of $(\sigma_j^{off}, St) \leftarrow$ **ReSign**$^{\mathbf{off}}$(**ReKey**$^{\mathbf{on/off}}(sk_i, sk_j, pk_i, pk_j)$, pk_i, pk_j); the on-line re-signing oracle $\mathcal{O}_{ReSign^{on}}(\cdot,\cdot,\cdot,\cdot)$ takes as input two distinct indices $i, j \in \{1, \ldots, n\}$, a message m and a signature σ_i, this oracle first retrieves the state information St from the memory, then returns the output of $\sigma_j^{on} \leftarrow$ **ReSign**$^{\mathbf{on}}$(**ReKey**$^{\mathbf{on/off}}(sk_i, sk_j, pk_i, pk_j), St, pk_i, m, \sigma_i$); and $\mathcal{Q}$ is defined as the set of (index, messages) pairs (i, m) queried the oracle $\mathcal{O}_{Sign}(\cdot,\cdot)$ or $\mathcal{O}_{ReSign^{on}}(\cdot,\cdot,\cdot,\cdot)$. This security notion only makes sense if the re-signing key is kept private by the proxy.

Internal Security: This security notion protects a user, as much as possible, from adversaries inside the system, such as dishonest proxy and colluding delegation partners. It can be classified into the following three types:

1. **Limited Proxy Security:** This security notion guarantees that the proxy cannot sign messages on behalf of the delegatee or produce signatures for the delegator unless messages were first signed be the latter's delegatees. Limited proxy security is very similar to external security described above except that the adversary queries a re-signature key generation oracle $\mathcal{O}_{ReKey^{on/off}}$ instead of the re-signing oracle (including $\mathcal{O}_{ReSign^{off}}$ and $\mathcal{O}_{ReSign^{on}}$). A divisible on-line/off-line proxy re-signature scheme is said to be limited-proxy-secure if for the security parameter k, any non-zero n, and all probabilistic polynomial-time algorithms $\mathcal{A}$, the following probability is negligible:

$$\begin{aligned}&\Pr[\{(pk_i, sk_i) \leftarrow \mathbf{KeyGen}(1^k)\}_{i\in\{1,\ldots,n\}},\\ &(i^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{Sign}(\cdot,\cdot),\mathcal{O}_{ReKey^{on/off}}(\cdot,\cdot)}(\{pk_i\}_{i\in\{1,\ldots,n\}}):\\ &\mathbf{Ver}(pk_{i^*}, m^*, \sigma^*) = 1 \wedge (i^*, m^*) \notin \mathcal{Q}]\end{aligned}$$

where $\mathcal{O}_{Sign}(\cdot,\cdot)$ is the same as that in external security; the re-signature key generation oracle $\mathcal{O}_{ReKey^{on/off}}(i, j)$ takes as input two distinct indices $i, j \in \{1, \ldots, n\}$ and returns the output of $RSK_{i\to j} \leftarrow$ **ReKey**$^{\mathbf{on/off}}$ (sk_i, sk_j, pk_i, pk_j); and $\mathcal{Q}$ is the set including any message m corresponding to which $\mathcal{A}$ has queried $\mathcal{O}_{Sign}(\diamond, m)$ for $\diamond = i^*$ or any $\diamond$ where $\mathcal{O}_{ReKey^{on/off}}(\cdot, \diamond)$ has been queried.

2. **Delegatee Security:** This notion protects the delegatee from a collusion between the delegator and the proxy. Delegatee security guarantees that their collusion cannot generate any signatures on behalf of the delegatee. Namely, the delegatee is assigned the index 0. A divisible on-line/off-line proxy re-signature scheme is said to be delegatee-secure if for the security parameter k, any non-zero n, and all probabilistic polynomial-time algorithms $\mathcal{A}$, the following probability is negligible:

$$\begin{aligned}&\Pr[\{(pk_i, sk_i) \leftarrow \mathbf{KeyGen}(1^k)\}_{i\in\{0,\ldots,n\}},\\ &(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{Sign}(\cdot,\cdot),\mathcal{O}_{ReKey^{on/off}}(\cdot,\Diamond)}(pk_0, \{pk_i, sk_i\}_{i\in\{1,\ldots,n\}}):\\ &\mathbf{Ver}(pk_0, m^*, \sigma^*) = 1 \wedge m^* \notin \mathcal{Q}]\end{aligned}$$

where $\Diamond \neq 0$, $\mathcal{O}_{Sign}(\cdot,\cdot)$ is the same as that in external security, $\mathcal{O}_{ReKey^{on/off}}(\cdot,\cdot)$ is the same as that in limited proxy security, and $\mathcal{Q}$ is the set of messages m queried to $\mathcal{O}_{Sign}(0, m)$.

3. **Delegator Security:** This notion guarantees that the collusion between the delegatee and the proxy cannot generate any original signatures on behalf of the delegator. Namely, we consider a target delegator with index 0. A divisible on-line/off-line proxy re-signature scheme is said to be delegator-secure if for the security parameter k, any non-zero n, and all probabilistic polynomial-time algorithms $\mathcal{A}$, the following probability is negligible:

$$\begin{aligned}&\Pr[\{(pk_i, sk_i) \leftarrow \mathbf{KeyGen}(1^k)\}_{i\in\{0,\ldots,n\}},\\ &(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{Sign}(\cdot,\cdot),\mathcal{O}_{ReKey^{on/off}}(\cdot,\cdot)}(pk_0, \{pk_i, sk_i\}_{i\in\{1,\ldots,n\}}):\\ &\mathbf{Ver}(pk_0, m^*, \sigma^*) \wedge m^* \notin \mathcal{Q}]\end{aligned}$$

where σ^* is an original signature, $\mathcal{O}_{Sign}(\cdot,\cdot)$ is the same as that in external security, $\mathcal{O}_{ReKey^{on/off}}(\cdot,\cdot)$ is the same as that in limited proxy security, and $\mathcal{Q}_{dtr}$ is the set of messages m queried to $\mathcal{O}_{Sign}(0, m)$.

Since the delegatee and the delegator mutually delegate in a bidirectional scheme, the properties of delegatee security and delegator security do not apply. We give the following security notions.

Definition 5. A bidirectional divisible on-line/off-line proxy re-signature scheme is secure if it is external-secure and limited-proxy-secure.

Definition 6. A unidirectional divisible on-line/off-line proxy re-signature scheme is secure if it is external-secure, limited-proxy-secure, delegatee-secure and delegator-secure.

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

In this section, we present a generic divisible on-line/off-line proxy re-signature scheme, and we show how the existing proxy re-signature scheme can be used in performing the computation of the off-line re-signature token. The proposed scheme is proven secure without random oracles. We assume that the message space is $\mathbb{Z}_q$. Note that using a collision-resistant hash function $H : \{0, 1\}^* \to \mathbb{Z}_q$, one can extend the message domain to $\{0, 1\}^*$.

4.1 Construction

Let **PRS**=(**KeyGen**, **ReKey**, **Sign**, **ReSign**, **Verify**) be a proxy re-signature scheme. The resulting divisible on-line/off-line proxy re-signature scheme is defined as **DOPRS** = (**KeyGen**, **ReKey$^{\mathbf{on/off}}$**, **Sign**, **ReSign$^{\mathbf{off}}$**, **ReSign$^{\mathbf{on}}$**, **Ver**), where

–**KeyGen** : Choose two primes p, q such that $q|p-1$, where p's length depends on the security parameter $k \in \mathbb{N}$. Pick an element g of order q in $\mathbb{Z}_p^*$. Let G be the subgroup of $\mathbb{Z}_p^*$ generated by g. Run the key generation algorithm **KeyGen** of the original proxy re-signature scheme **PRS** to generate signer's secret/public key pair (sk, pk). Let cp be the set of the public parameters of **PRS**. The proxy chooses at random $y, z \in \mathbb{Z}_q^*$, and computes $h_1 = g^y$, $h_2 = g^z$ and $Y = y^{-1} (\bmod\ q)$. The set of the public parameters is $params$:=$\{cp, p, q, g, h_1, h_2\}$.

–**ReKey$^{\mathbf{on/off}}$** : On input an (optional) delegatee's private key sk_A, a delegator's private key sk_B, and the corresponding public keys (pk_A, pk_B), the proxy first runs the re-signature key generation algorithm **ReKey** of **PRS** with (sk_A, sk_B, pk_A, pk_B) to obtain a re-signature key $rk_{A\to B}$ for **PRS**. The proxy then retains $RSK_{A\to B} = (rk_{A\to B}, y, z, Y)$ as its own local re-signature key for **DOPRS**.

–**Sign** : On input a secret signing key sk and a message m', it runs the signing algorithm **Sign** of **PRS** with sk to obtain a signature σ=**Sign**(sk, m').

–**ReSign$^{\mathbf{off}}$** : In order to generate an off-line re-signature token σ_B^{off}, this algorithm runs as follows:

1.The proxy chooses at random $m, r, s, s' \in \mathbb{Z}_q^*$ and sends $Com = g^m h_1^r h_2^s$ to a delegatee.

2.The delegatee runs the algorithm **Sign** of **PRS** with sk_A to compute a signature σ_A on the message Com, and sends σ_A to the proxy.

3.The proxy checks **Verify**$(pk_A, Com, \sigma_A) \overset{?}{=} 1$. If the equation holds, the proxy runs the re-signing algorithm **ReSign** of **PRS** with $(rk_{A\to B}, Com, \sigma_A)$ to obtain a re-signature σ_B on Com.

4.The proxy computes $\tau = m + ry + (s - s')z(\bmod\ q)$, and stores the state information $St = \tau$. The off-line re-signature token $\sigma_B^{off} = (pk_A, \sigma_B)$.

–**ReSign$^{\mathbf{on}}$** : On input a re-signature key $RSK_{A\to B}$, a delegatee's public key pk_A, a message m' and a signature σ_A', the proxy first retrieves the state information St from the memory, and then computes $r' = (St - m')Y(\bmod\ q)$. The on-line re-signature token on m' is $\sigma_B^{on} = (r', s', \sigma_A')$. The re-signature for m' is given by $\sigma_B' = (\sigma_B^{off}, \sigma_B^{on}) = (\sigma_{B,0}, \sigma_{B,1}, \sigma_{B,2}, \sigma_{B,3}, \sigma_{B,4}) = (pk_A, \sigma_B, r', s', \sigma_A')$.

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

–If σ is an original signature under pk, it runs the verification algorithm **Verify** of **PRS** to check that **Verify**$(pk, m', \sigma) \overset{?}{=} 1$. If this equation holds, it outputs 1; otherwise, it outputs 0.

–If σ is a re-signature $\sigma = (\sigma_0, \sigma_1, \sigma_2, \sigma_3, \sigma_4)$ under pk, it checks that

$$\mathbf{Verify}(pk, g^{m'} h_1^{\sigma_2} h_2^{\sigma_3}, \sigma_1) \overset{?}{=} 1 \quad \text{and}$$

$$\mathbf{Verify}(\sigma_0, m', \sigma_4) \overset{?}{=} 1 .$$

If the above two equations both hold, it outputs 1; otherwise, it outputs 0.

Correctness: Let σ_A' be an original signature on a message m' corresponding to a delegatee's public key pk_A and σ_B be a re-signature on $Com = g^m h_1^r h_2^s$ under a delegator's public key pk_B. Let $\sigma_B' = (\sigma_B^{off}, \sigma_B^{on}) = (\sigma_{B,0}, \sigma_{B,1}, \sigma_{B,2}, \sigma_{B,3}, \sigma_{B,4}) = (pk_A, \sigma_B, r', s', \sigma_A')$ be a re-signature on the same message m' under pk_B. The proposed scheme has the correctness due to the following equations.

$$\begin{aligned} g^{m'} h_1^{\sigma_{B,2}} h_2^{\sigma_{B,3}} &= g^{m'} (g^y)^{r'} (g^z)^{s'} = g^{m' + yr' + zs'} \\ &= g^{m' + y(St - m')Y + zs'} = g^{m' + (\tau - m')yY + zs'} \\ &= g^{m' + m + ry + (s - s')z - m' + zs'} = g^{m + ry + sz} = Com \end{aligned}$$

$$\begin{aligned} &\mathbf{Verify}(pk_B, g^{m'} h_1^{\sigma_{B,2}} h_2^{\sigma_{B,3}}, \sigma_{B,1}) \\ &\qquad = \mathbf{Verify}(pk_B, Com, \sigma_B) = 1 \end{aligned}$$

$$\mathbf{Verify}(\sigma_{B,0}, m', \sigma_{B,4}) = \mathbf{Verify}(pk_A, m', \sigma_A') = 1$$

Remark 3: Sign, **ReSign$^{\mathbf{off}}$**, **ReSign$^{\mathbf{on}}$** and **Ver** are performed per message.

Remark 4: To reduce the computation cost of the on-line re-signing algorithm, we move the validity of the original signature σ_A' to the verification algorithm **Ver**. As a result, σ_A' is attached as part of the on-line re-signature token. To save the on-line bandwidth, the transmission of the whole re-signature $\sigma_B' = (\sigma_B^{off}, \sigma_B^{on})$ is broken into two stages. The on-line re-signature token $\sigma_B^{on} = (\sigma_{B,2}, \sigma_{B,3}, \sigma_{B,4}) = (r', s', \sigma_A')$ on a message m' is transmitted to the recipient at the end of the on-line phase, while the off-line re-signature token $\sigma_B^{off} = (\sigma_{B,0}, \sigma_{B,1}) = (pk_A, \sigma_B)$ is transmitted in the off-line phase.

Remark 5: If the underlying proxy re-signature scheme **PRS** is bidirectional, the re-signature key $rk_{A\to B}$ can be used to transform delegatee's signatures into delegator's or vice versa. If the underlying scheme **PRS** is unidirectional, $rk_{A\to B}$ can be used to transform delegatee's signatures into delegator's, but not vice versa. Moreover, (y, z, Y) are three secret trapdoor keys of the chameleon hash function $g^m h_1^r h_2^s$. Hence, the proposed divisible on-line/off-line proxy re-signature scheme **DOPRS** with $RSK_{A\to B} = (rk_{A\to B}, y, z, Y)$ is bidirectional (resp. unidirectional) if so is its underlying proxy re-signature scheme **PRS**.

4.2 Security

The following theorem shows that the security of the proposed divisible on-line/off-line proxy re-signature scheme is reduced to the security of its underlying proxy re-signature scheme together with computational infeasibility of the discrete logarithm assumption. In essence, two schemes hold the same security properties. For example, if an underlying bidirectional proxy re-signature scheme is external-secure and limited-proxy-secure, the resulting divisible on-line/off-line proxy re-signature scheme is also external-secure and limited-proxy-secure.

Theorem 1*Assuming that* **PRS**=(**KeyGen**, **ReKey**, **Sign**, **ReSign**, **Verify**) *is a secure proxy re-signature scheme and the discrete logarithm assumption holds, our divisible on-line/off-line proxy re-signature scheme* **DOPRS** = (**KeyGen**, **ReKey$^{\mathbf{on/off}}$**, **Sign**, **ReSign$^{\mathbf{off}}$**, **ReSign$^{\mathbf{on}}$**, **Ver**) *is also secure.*

Proof. We prove this theorem by contradiction. Assume that there exists an adversary $\mathscr{A}$ that breaks the unforgeability of our scheme **DOPRS** and we show how to exploit it to break either the unforgeability of the underlying proxy re-signature scheme **PRS** or the discrete logarithm assumption. In other words, we can construct an efficient algorithm $\mathscr{B}$ that, using $\mathscr{A}$ as a black box, succeeds in above mentioned tasks.

Without loss of generality, we assume that $\mathscr{A}$ makes q_s signature queries, q_{off} off-line re-signature queries and q_{on} on-line re-signature queries on messages $\{m_i^{'}\}_{i\in\{1,\cdots,q_{on}\}}$, where $q_{on} \leq q_{off}$ and $q_{on} \leq q_s$. Let $\{(\sigma_{i,0}^{'}, \sigma_{i,1}^{'}, r_i^{'}, s_i^{'}, \sigma_{i,4}^{'})\}_{i\in\{1,\cdots,q_{on}\}}$ be q_{on} full re-signatures returned by the re-signing oracle (including the off-line and on-line re-signing oracles). Finally, $\mathscr{A}$ outputs a valid forgery σ^* on a new message m^* with probability ε. If σ^* is an original signature, then σ^* is also a valid forgery of the underlying proxy re-signature scheme **PRS**. Hence, for simplicity, we assume that σ^*=(σ_0^*, σ_1^*, r^*, s^*, σ_4^*) is a re-signature. Notice that, any valid forgery must be one of the following types:

Case 1: $g^{m^*}h_1^{r^*}h_2^{s^*} \neq g^{m_i^{'}}h_1^{r_i^{'}}h_2^{s_i^{'}}$ for all $i \in \{1,\cdots,q_{on}\}$.

Case 2: $g^{m^*}h_1^{r^*}h_2^{s^*} = g^{m_i^{'}}h_1^{r_i^{'}}h_2^{s_i^{'}}$ for some $i \in \{1,\cdots,q_{on}\}$ and $r^* \neq r_i^{'}$.

Case 3: $g^{m^*}h_1^{r^*}h_2^{s^*} = g^{m_i^{'}}h_1^{r_i^{'}}h_2^{s_i^{'}}$ for some $i \in \{1,\cdots,q_{on}\}$, $r^* = r_i^{'}$ and $s^* \neq s_i^{'}$.

If the first case holds, we build an algorithm $\mathscr{B}$ against the unforgeability of **PRS** with probability at least $\varepsilon/3$. $\mathscr{B}$ is given the set cp of the public parameters of **PRS**. His goal is to use the forgery produced by the adversary $\mathscr{A}$ to contradict the existential unforgeability of **PRS**. $\mathscr{B}$ works as follows:

Setup: $\mathscr{B}$ chooses two random numbers y, z from $\mathbb{Z}_p^*$, and computes $h_1 = g^y, h_2 = g^z$. The public key of the target delegatee is defined as pk_A and the public key of the target delegator is defined as pk_B. Then, $\mathscr{B}$ sends $\{cp, p, q, g, pk_A, pk_B, h_1, h_2\}$ to $\mathscr{A}$. When $\mathscr{A}$ issues a key generation query, $\mathscr{B}$ first forward this query to the challenger of **PRS**; the challenger returns the resulting public key or secret/public key pair to $\mathscr{B}$; $\mathscr{B}$ then sends it to $\mathscr{A}$.

Queries: $\mathscr{B}$ builds the following oracles:

–Re-signature key generation oracle $\mathscr{O}_{ReKey^{on/off}}$: When $\mathscr{A}$ issues a re-signature key generation query which consists of two public keys (pk_i, pk_j), $\mathscr{B}$ submits the query (pk_i, pk_j) to the re-signature key generation oracle of **PRS** to obtain a corresponding re-signature key $rk_{i\to j}$, and then $\mathscr{B}$ returns a re-signature key $RSK_{i\to j} = (rk_{i\to j}, y, z, z^{-1})$ to $\mathscr{A}$.

–Signature oracle $\mathscr{O}_{Sign}$: When $\mathscr{A}$ issues a signing query which consists of a public key pk_i and a message $m^{'}$, $\mathscr{B}$ submits the query $(pk_i, m^{'})$ to the signing oracle of **PRS** to obtain a corresponding signature σ and sends σ to $\mathscr{A}$.

–Off-line re-signature oracle $\mathscr{O}_{ReSign^{off}}$: On input (pk_i, pk_j), $\mathscr{B}$ first chooses $m_i, r_i, s_i, s_i^{'} \in \mathbb{Z}_p^*$ and computes $Com_i = g^{m_i}h_1^{r_i}h_2^{s_i}$. $\mathscr{B}$ then queries the signing oracle of **PRS** to get a signature $\sigma_{A,i}$ on Com_i under the public key pk_i. Furthermore, $\mathscr{B}$ queries the re-signing oracle of **PRS** providing $(pk_i, pk_j, Com_i, \sigma_{A,i})$ as input and obtains a resulting re-signature $\sigma_{B,i}$. Finally, $\mathscr{B}$ returns an off-line re-signature token $\sigma_j^{off} = (pk_i, \sigma_{B,i})$ to $\mathscr{A}$ and stores the secret state information $(m_i, r_i, s_i, s_i^{'})$.

–On-line re-signature oracle $\mathscr{O}_{ReSign^{on}}$: On input $(pk_j, pk_i, m_i^{'}, \sigma_i^{'})$, $\mathscr{B}$ outputs $\perp$ if $\mathbf{Ver}(pk_i, m_i^{'}, \sigma_i^{'}) = 0$. Otherwise, $\mathscr{B}$ computes $r_i^{'} = r_i + (m_i + zs_i - m_i^{'} - zs_i^{'})y^{-1}(\mathrm{mod}\ q)$, and then returns an on-line re-signature token $\sigma_j^{on} = (r_i^{'}, s_i^{'}, \sigma_i^{'})$. It can be verified that $(pk_i, \sigma_{B,i}, r_i^{'}, s_i^{'}, \sigma_i^{'})$ is a valid signature on the message $m_i^{'}$ under the public key pk_j.

Forgery: The simulated oracles are perfectly indistinguishable from the real ones for $\mathscr{A}$. At the end, $\mathscr{A}$ outputs a valid forgery σ^*=(σ_0^*, σ_1^*, r^*, s^*, σ_4^*) on a message m^* satisfying the condition in Case 1. This means that m^* was never queried by $\mathscr{B}$ to the signing oracle and the re-signing oracle of **PRS**. But in this case, $\mathscr{B}$ has forged a signature σ_1^* on a message $g^{m^*}h_1^{r^*}h_2^{s^*}$ with respect to the underlying proxy re-signature scheme **PRS**. Thus, $\mathscr{B}$ succeeds in breaking the unforgeability of the the underlying proxy re-signature scheme.

[CASE 2.]

If the second case holds, we build an algorithm $\mathscr{B}$ that breaks the discrete logarithm assumption. To do so, $\mathscr{B}$ is given a couple $(g, h) \in G^2$. His goal is to use the forgery produced by $\mathscr{A}$ to compute the discrete log $dl_g h$ of h in base g. $\mathscr{B}$ works as follows:

Setup: $\mathscr{B}$ first chooses two random elements $\hat{y}, z \in \mathbb{Z}_p^*$, and sets $h_1 = h$, $h_2 = g^z$. The other public parameters are same as those in the first case. Note that $\hat{y}$ differs from y. Then, $\mathscr{B}$ sends $\{cp, p, q, g, pk_A, pk_B, h_1, h_2\}$ to $\mathscr{A}$.

Queries: $\mathscr{B}$ builds the following oracles:

–$\mathscr{O}_{Sign}$ is the same as that in the first case.
–Re-signature key generation oracle $\mathscr{O}_{ReKey^{on/off}}$: When $\mathscr{A}$ issues a re-signature key generation query which consists of two public keys (pk_i, pk_j), $\mathscr{B}$ submits the query (pk_i, pk_j) to the re-signature key generation oracle of **PRS** to obtain a corresponding re-signature key $rk_{i\to j}$, and then $\mathscr{B}$ returns a re-signature key $RSK_{i\to j} = (rk_{i\to j}, \hat{y}, z, z^{-1})$ to $\mathscr{A}$.
–Off-line re-signature oracle $\mathscr{O}_{ReSign^{off}}$: On input (pk_i, pk_j), $\mathscr{B}$ first chooses $m_i, r_i, s_i \in \mathbb{Z}_p^*$, sets $s_i' = (s_i + (m - m')z^{-1}) (\text{mod } q)$, and computes $Com_i = g^{m_i} h_1^{r_i} h_2^{s_i}$. $\mathscr{B}$ then queries the signing oracle of **PRS** to get a signature $\sigma_{A,i}$ on Com_i under the public key pk_i. Furthermore, $\mathscr{B}$ queries the re-signing oracle of **PRS** providing $(pk_i, pk_j, Com_i, \sigma_{A,i})$ as input and obtains a resulting re-signature $\sigma_{B,i}$. Finally, $\mathscr{B}$ returns an off-line re-signature token $\sigma_j^{off} = (pk_i, \sigma_{B,i})$ to $\mathscr{A}$ and stores the secret state information (m_i, r_i, s_i, s_i').
–On-line re-signature oracle $\mathscr{O}_{ReSign^{on}}$: On input $(pk_j, pk_i, m_i', \sigma_i')$, $\mathscr{B}$ outputs $\perp$ if $\mathbf{Ver}(pk_i, m_i', \sigma_i') = 0$. Otherwise, $\mathscr{B}$ computes $r_i' = r_i$, and then returns an on-line re-signature token $\sigma_j^{on} = (r_i', s_i', \sigma_i')$. It can be verified that $(pk_i, \sigma_{B,i}, r_i', s_i', \sigma_i')$ is a valid signature on the message m_i' under the public key pk_j.

Forgery: From $\mathscr{A}$'s view, the simulated oracles are indistinguishable from the real ones. If $\mathscr{A}$ outputs a valid forgery $\sigma^*=(\sigma_0^*, \sigma_1^*, r^*, s^*, \sigma_4^*)$ on a message m^* satisfying the condition in Case 2, then $g^{m^*} h_1^{r^*} h_2^{s^*} = g^{m_i'} h_1^{r_i'} h_2^{s_i'}$ and $r^* \neq r_i'$ for some i hold. $\mathscr{B}$ can check to find this i and obtain the discrete log $dl_g h = ((m_i' - m^*) + (s_i' - s^*)z)(r^* - r_i')^{-1} (\text{mod } q)$. Thus, $\mathscr{B}$ can successfully solve the received discrete logarithm instance.
[CASE 3.]

If the third case holds, we construct an algorithm $\mathscr{B}$ that breaks the discrete logarithm assumption. To do so, $\mathscr{B}$ is given a couple $(g, h) \in G^2$. $\mathscr{B}$'s goal is to compute the discrete log $dl_g h$ of h in base g. We focus on describing the differences with the second case.
Setup: $\mathscr{B}$ chooses two random elements $y, \hat{z} \in \mathbb{Z}_p^*$, and sets $h_1 = g^y$, $h_2 = h$. Thus, $\mathscr{B}$ knows y and a value $\hat{z}$ that differs from z.
Queries: $\mathscr{B}$ builds the following oracles:

–$\mathscr{O}_{ReKey^{on/off}}$ and $\mathscr{O}_{Sign}$ are identical to those in Case 2, by just switching the roles of z and y.
–Off-line re-signature oracle $\mathscr{O}_{ReSign^{off}}$: On input (pk_i, pk_j), $\mathscr{B}$ chooses $m_i, r_i, s_i \in_R \mathbb{Z}_p^*$, sets $s_i' = s_i$, and computes $Com_i = g^{m_i} h_1^{r_i} h_2^{s_i}$. The rest of the oracle is done exactly as in Case 2.

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

Schemes	Subtractions	Multiplications	Exponentiations	Pairings
$\mathbf{S_{bi}}$	0	0	1	2
$\mathbf{S_{uni}}$	0	1	3	2
$\mathbf{S_{mb}}$	0	0	2	3
CP – PRS	0	3	2	2
LV – PRS	0	2	6	3
Our **DOPRS**	1	1	0	0

–On-line re-signature oracle $\mathscr{O}_{ReSign^{on}}$: On input $(pk_j, pk_i, m_i', \sigma_i')$, $\mathscr{B}$ outputs $\perp$ if $\mathbf{Ver}(pk_i, m_i', \sigma_i') = 0$. Otherwise, $\mathscr{B}$ computes $r_i' = r_i + (m_i - m_i')y^{-1} (\text{mod } q)$, and then returns an on-line re-signature token $\sigma_j^{on} = (r_i', s_i', \sigma_i')$.

Forgery: If $\mathscr{A}$ outputs a valid forgery $\sigma^*=(\sigma_0^*, \sigma_1^*, r^*, s^*, \sigma_4^*)$ on a message m^* satisfying the condition in Case 3, then $g^{m^*} h_1^{r^*} h_2^{s^*} = g^{m_i'} h_1^{r_i'} h_2^{s_i'}$ and $s^* \neq s_i'$ for some i hold. $\mathscr{B}$ then can easily get the discrete log $dl_g h = ((m_i' - m^*) + (r_i' - r^*)y)(s^* - s_i')^{-1} (\text{mod } q)$. Therefore, $\mathscr{B}$ succeeds in breaking the received discrete logarithm challenge.

4.3 Comparison

We compare our divisible on-line/off-line proxy re-signature scheme with some known proxy re-signature schemes. We mainly analyze bit complexity of all schemes required by a proxy in computing a re-signature when a message to be re-signed arrives, and show the results in Table 1. However, schemes $\mathbf{S_{bi}}$ and $\mathbf{S_{uni}}$ presented by Ateniese and Hohenberger [2], scheme $\mathbf{S_{mb}}$ proposed by Shao *et al.* [12], scheme **CP – PRS** proposed by Chow and Phan [13], and scheme **LV – PRS** presented by Libert and Vergnaud [14] are not considered to be on-line/off-line proxy re-signature schemes because no pre-computation is performed. To achieve the same security level, we can set the modular parameter p in all schemes to be the same size.

From Table 1, we can see that the proxy performs one modular subtraction and one modular multiplication when computing a re-signature in our scheme. This is very efficient and comparable to the other schemes, since modular subtraction and modular multiplication are negligible when compared with exponentiation and pairing. Our scheme makes a tradeoff by incurring a large cost in the off-line phase to obtain a quick on-line phase.

5 Conclusion

We introduce a notion called divisible on-line/off-line proxy re-signatures, in which the off-line re-signature tokens can be sent to the recipient before the messages to be re-signed are seen. We also present an efficient generic construction, and prove its security without random oracles.

Acknowledgement

This research is supported in part by the National Natural Science Foundation of China (Grant Nos. 61262057, 61163038, 61063041 and 61202362), Natural Science Foundation of Gansu Province of China (Grant No. 1308RJYA039), 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-11-24). 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**, 127-144 (1998).
[2] G. Ateniese and S. Hohenberger, Proxy re-signatures: new definitions, algorithms and applications, Proceedings of ACM Conference on Computer and Communications Security, 310-319 (2005).
[3] P.Y. Yang, Z.F. Cao, and X. Dong, Threshold proxy re-signature, Journal of Systems Science and Complexity, **24**, 816-824 (2011).
[4] X.D. Yang and C.F. Wang, Threshold proxy rełsignature schemes in the standard model, Chinese Journal of Electronics, **20**, 691-696 (2011).
[5] X. Hong and Y. Long, A novel unidirectional proxy re-Signature scheme and its application for MANETs, Journal of Computers, **7**, 1796-1800 (2012).
[6] D.F. He, A novel blind proxy re-signature scheme, Computer Applications and Software, **29**, 294-296 (2012).
[7] S.S. Rawat and G.K. Shrivastava, Improved id-based proxy re-signcryption scheme, Proceedings of 2012 IEEE CICN, Mathura, India, 730-733 (2012).
[8] Y.Q. Deng and G. Song, Proxy re-signature scheme based on quadratic residues, Journal of Networks, **6**, 1459-1465 (2011).
[9] D.t. Guo, P. Wei, D. Yu, and X.Y Yang, A certificateless proxy re-signature scheme, Proceedings of IEEE ICCSIT 2010, Xi'an, China, 157-161 (2010).
[10] J. Shao, G. Wei, Y. Ling, M. Xie, Unidirectional identity-based proxy re-signature, Proceedings of 2011 IEEE ICC, Kyoto, Japan, 1-5 (2012).
[11] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, Secure distributed key generation for discrete-log based cryptosystems, Proceedings of EUROCRYPT'99, Springer: Berlin, 295-310 (1999).
[12] J. Shao, Z.F. Cao, L. Wang, and X. Liang, Proxy re-signature schemes without random oracles, Proceedings of INDO-CRYPT 2007, Chennai, India, 197-209 (2007).
[13] S.S.M. Chow and R.C.-W. Phan, Proxy re-signatures in the standard model, Proceedings of 11th International Conference on Information Security, Taibei, China, 260-276 (2008).
[14] B. Libert and D. Vergnaud, Multi-use unidirectional proxy re-signatures, Proceedings of the 15th ACM Conference on CCS, 511-520 (2008).
[15] H. Krawczyk and T. Rabin, Chameleon hashing and signatures, in Proceedings of the 7th Annual Network and Distributed System Security Symposium, 143-154 (2000).
[16] X. Yang, C. Wang, L. Zhang, et al. On-line/off-line threshold proxy re-signatures. Chinese Journal of Electronics, **23**, 248-253 (2014).
[17] Z. Wang, W. Chen. An ID-based online/offline signature scheme without random oracles for wireless sensor networks. Personal and ubiquitous computing, **17**, 837-841 (2013).
[18] E. Bresson, D. Catalano, M. Raimondo, et al. Off-line/on-line signatures revisited: a general unifying paradigm, efficient threshold variants and experimental results. International journal of information security, **12**, 439-465 (2013).
[19] C. Gao, B. Wei, D. Xie, and C. Tang, Divisible on-line/off-line signatures, Proceedings of CT-RSA **2009**, 562-566 (2009).
[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**, 216-232 (2010).

Xiaodong YANG is 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.

Chunmei Li is now a postgraduate student of cryptography at Northwest Normal University. Her current research interests include network security and cryptography. He is also a member of Chinese Cryptology and Information Security Association.

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

Sian Zhou is now a postgraduate student of cryptography at Northwest Normal University. His current research interests include information security.

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.