Threshold Proxy Re-signature Scheme with Privacy

Kuan-Yu Chen and Hsi-Chung Lin

***Abstract*—A proxy signature scheme allows a designated party to sign on behalf of the original signer. However, a dishonest proxy signer can manage to sign arbitrary messages and jeopardize the original signer's benefit. Alternatively, a proxy re-signature scheme allows the original signer to delegate his signing right in a decentralized manner, to a delegatee and a semi-trusted transformer. In order to protect the privacy of the delegatee, a proxy re-signature scheme with privacy consideration was recently introduced. In this paper, we integrate the idea of threshold delegation into a proxy re-signature scheme with privacy consideration and the resulting scheme is useful in a new delegating scenario of signing right delegation.**

***Index Terms*—Privacy, proxy re-signature, threshold delegation.**

I. Introduction

In the study of signing right delegation, a proxy signature scheme [1], [2] allows the original signer to delegate his signing right to a designated party, then the party can sign a signature on behalf of the original signer. However, a dishonest proxy signer can manage to sign arbitrary messages and damage the original signer's benefit. Hence, in earlier literatures, a lot of research works focus on the design of mechanisms to restrict the proxy signer's signing ability.

On the other hand, in some recent literatures [3]- [5], proxy re-signature schemes are widely discussed. A proxy re-signature scheme allows the original signer (a.k.a., the *delegator*) to delegate his signing right to a designated participant (a.k.a., the *delegatee*), to sign signatures on his behalf with the help of a semi-trusted party known as a *transformer*. In brief, the delegatee signs a message with his secret key first, then the signature will be transformed to a signature of the delegator (on the same message) by the semi-trusted transformer.

Generally, the signing right is decentralized in a proxy re-signature scheme and the delegatee is not capable of signing on the delegator's behalf independently; nevertheless, a malicious delegatee can still sign arbitrary messages since the semi-trusted party transforms naively in most proxy re-signature scheme in the literature.

To counter this potential weakness, an advanced proxy re-signature scheme with warrant is proposed in [6]. In that scheme, the semi-trusted party is empowered to decide whether or not he should transform the signature; hence the signing right of the delegator is more rigorously protected. Besides, in that scheme, a signature signed by the delegatee is not publicly verifiable; hence the privacy of the delegatee is well protected.

In the aforementioned proxy re-signature scheme with warrant and privacy consideration, the delegator delegates his signing right to a single delegatee, with the cooperation of the semi-trusted transformer. It is not applicable in the application scenario involving multiple delegatees. In this paper, we integrate the secret sharing mechanism introduced by Shamir [7] with proxy re-signatures, and propose a (t, n) threshold proxy re-signature scheme with privacy. The proposed scheme can be applied in the new delegating scenario of multiple delegatees and one semi-trusted transformer. Besides, in comparison with schemes in [3]-[5], the delegator in the proposed scheme delegates his signing right in a non-interactive manner; and a signature generated by a delegatee is not publicly verifiable, but is designated verifiable.

Manuscript received October 20, 2012; revised November 24, 2012.

Kuan-Yu Chen and Hsi-Chung Lin are with the Department of Computer Science and Information Engineering, Aletheia University, No.32, Zhenli St., Danshui Dist., New Taipei City 25103, Taiwan, R.O.C. (e-mail: fm990055@ mail.au.edu.tw; hclin@ mail.au.edu.tw).

II. Preliminaries

A. Proxy Re-signature Scheme

In brief, a proxy re-signature scheme allows a semi-trusted party to transform a signature of a participant to a signature on exactly the same message of another party. When applied to the scenario of signing right delegation, it enables an original signer (the delegator) to delegate his signing right to a transformer and a delegatee, then it allows the delegatee to sign on behalf of the delegator, with the help of the transformer.

The idea of proxy re-signature scheme was first introduced in [3] and was further improved in [4]. Potential weakness of [4] is pinpointed and new proxy re-signature schemes were suggested in [5]. Basically, a proxy re-signature scheme consists of the following algorithms.

- **KeyGen**: The key generation algorithm generates necessary key-pairs for participants.
- **ReKeyGen**: The resigning key generation algorithm outputs a resigning key for the semi-trusted party to translate a signature of one participant to a signature of another party.
- **Sign**: The delegatee uses this function to generate a signature which will be translated later.
- **Resign**: With the resigning key, the semi-trusted party launches this function and translates the signature of the delegatee into a signature of the delegator.
- **Verify**: The verification algorithm verifies the validity of the translated signature.

B. The Proxy Re-signature Scheme with Privacy

Our new scheme is based on the proxy re-signature scheme with privacy considerations [6]. It requires a bilinear map e:$G_1 \times G_1 \rightarrow G_2$ which operates over group G_1 and G_2 of

prime order $p = \Theta(2^k)$ with the following properties.

1) For all $a, b \in Z_q$, $g \in G_1$, $e(g^a,g^b) = e(g^b,g^a) = e(g,g)^{ab} \in G_2$.
2) The map is non-degenerate.

Global parameters include (e, q, G_1, G_2, g, h, H), where g generates G_1, h is a hash function from an arbitrary string to an element of Z_q, and H is a map-to-point hash function from an arbitrary string to an element of G_1.

- **KeyGen**(1^k): On input of the security parameter 1^k, it generates the public/private key pair (Y_l, x_l) for participant l ($l \in \{A, B, T\}$) where $Y_l = g^{x_l}$ for some random $x_l \in Z_q$.
- **Delegate**(x_A, Y_B, z): On input of the delegatee's public key Y_B, and the secret value z shared with the transformer, the delegator sets a warrant message w and generates a pair (R, S) for a delegatee of this choice

$$R = Y_A^{\beta} \ ; \ S = g^{x_A^{-1}\beta}$$

where $\beta = h(Y_B \| w \| z)$. This pair should be sent to the delegatee via a confidential channel. With this pair at hand, the delegatee can generate a signature which will be transformed to a signature of the delegator later.

- **Sign**(m, x_B, R, S): To sign (this is a first level signature following the definition of [5]) a message m, the delegatee uses his private key x_B and the pair (R, S) the delegator provided to compute $\sigma_B = (\sigma_B^{(0)}, \sigma_B^{(1)})$ where

$$\sigma_B^{(0)} = H(m)^{x_B} \cdot R \ ;$$
$$\sigma_B^{(1)} = S^{x_B}.$$

- **Resign**(σ_B, β): On input (σ_B, β), the transformer T first verifies the validity of σ_B by the following two equations

$$e(\sigma_B^{(0)}, g) = e(H(m), Y_B) \cdot e(Y_A, g)^{\beta};$$
$$e(\sigma_B^{(1)}, Y_A) = e(Y_B, g)^{\beta}.$$

If both equations hold, the transformer selects a random value $\alpha \in Z_q$, and outputs $\sigma_A = (\sigma_A^{(0)}, \sigma_A^{(1)}, \sigma_A^{(2)})$ where

$$\sigma_A^{(0)} = (\sigma_B^{(1)} \cdot Y_A^{-\beta})^{\alpha};$$
$$\sigma_A^{(1)} = Y_B^{\alpha}; \ \sigma_A^{(2)} = (\sigma_B^{(1)})^{\beta^{-1}\alpha}.$$

- **Verify**(σ_A, m, Y_A): On input the delegator's public key Y_A, the message m, and the signature σ_A, anyone can verify the validity of the signature by the following equations

$$e(\sigma_A^{(0)}, g) = e(H(m), \sigma_A^{(1)});$$
$$e(\sigma_A^{(1)}, g) = e(\sigma_A^{(2)}, Y_A).$$

The algorithm outputs accept if both equation hold, and outputs reject otherwise.

Please remark that there are two more algorithms in this scheme, but they are not explicitly used in most cases.

Registration(x_A, x_T): The delegator must register with the transformer before he can delegates his signing right to any delegatee. The main objective of this process is to generate a shared secret z between the delegator and the transformer by which allows them to compute the secret value β. This processes is implicit in most cases, for example we can sets $z = Y_A^{x_T} = Y_T^{x_A}$ which is the Diffie-Hellman key of these two participants.

- **ASign**(m, x_A): To sign (this is second level signature following the definition of [5]) a message m directly by himself, the delegator selects at random $t \in Z_q$ and outputs $\sigma_{\tilde{A}} = (\sigma_{\tilde{A}}^{(0)}, \sigma_{\tilde{A}}^{(1)}, \sigma_{\tilde{A}}^{(2)})$ where

$$\sigma_{\tilde{A}}^{(0)} = H(m)^{x_A t};$$
$$\sigma_{\tilde{A}}^{(1)} = Y_A^{t};$$
$$\sigma_{\tilde{A}}^{(2)} = g^{t}.$$

Notice that the signature $\sigma_{\tilde{A}}$ can pass the verification algorithm. Moreover, σ_A and $\sigma_{\tilde{A}}$ should be computationally indistinguishable.

C. Properties Related to Proxy Re-signature

Properties related to the proposed scheme are reviewed here, most of them are discussed in [3]-[5]. Since the proposed scheme can be put into practice in a new scenario of signing right delegation, some of the properties are modified and new properties are also added.

1) *Unidirectional*: In a unidirectional scheme, the semi-trusted party translates a signature in only one direction; he transforms the delegatee's signature into the delegator's signature but not vice versa.
2) *Single-use*: In a single-use proxy re-signature scheme, a transformed signature cannot be transformed again.
3) *Private Proxy*: In a scheme with private proxy, the re-signing key is kept secret.
4) *Transparent*: A signature generated directly by the delegator and a signature translated by the semi-trusted party are indistinguishable for observers.
5) *Unlinkable*: A translated signature cannot be linked to the one from which it was generated.
6) *Non-transitive*: The semi-trusted party cannot re-delegate signing rights.
7) (*t,n*) *threshold*: The delegator delegates his signing right to n delegatees. It requires at least t delegatees and a transformer to generate a signature on behalf of the delegator.
8) *Designated verifiable*: A signature generated by a delegatee is not public verifiable. Instead, it is a strong designated verifier signature [8] which can only be verified by the transformer.

III. Threshold Proxy Re-Signature with Privacy

In the proposed scheme, the delegator A delegates his signing right to n delegatee B_i for $i = 1,2,...,n$; a valid signature of the delegator A can be produced cooperatively by greater than or equal to t delegatees and a semi-trusted transformer T. In other words, in a threshold proxy re-signature scheme, the transformer T has to collect at least t delegatees' signatures on the same message, combine them,

and then translate it to a signature of the delegator A on the same message.

Definition 1: A (t, n) threshold proxy re-signature with privacy consists of the following polynomial time algorithms: KeyGen, Registration, ShareReKey, CombineSign, ASign, CombineReSign, and Verify.

- **KeyGen**: The key generation algorithm generates necessary public/private key pairs for all participants.
- **Registration**: The delegator A should register the service provided by the transformer before he can delegate his signing right to n delegates B_i. Basically this is a step for the delegator A and the transformer T to generate a shared secret z; this step might be executed implicitly since the secret z can be generated easily by Diffie-Hellman key exchange if both of them have public/private key pairs.
- **ShareReKey**: Using this algorithm, the delegator A computes necessary values for all delegatees B_i and delivers those values to all delegatees in a secure way. Those values enable delegatees to sign on the delegator's behalf.
- **CombineSign**: In this function, a delegatee B_i has to cooperate with at least $t-1$ delegatees, and uses his own private signing key to generate his own signature, which is an untransformed signature and is only verifiable to the transformer. In the cooperative part of the process, a delegatee B_i securely exchanges respective secret values with other $t-1$ delegatees who also agree to sign the message.
- **ASign**: Besides delegating his signing right to n delegates B_i, the delegator A himself should be capable of signing message, and the signature he generated should be indistinguishable to the one produced by (at least) t delegatees B_i and the transformer T cooperatively.
- **CombineReSign**: The transformer T receives at least t untransformed signatures. He combines those untransformed signatures and translates them into the delegator's signature if those signatures can pass verifications.
- **Verify**: Anyone can uses this algorithm to verify the validity of delegator A's signatures.

A. The Proposed Scheme

The proposed scheme requires a bilinear map, where the map $e: G_1 \times G_1 \rightarrow G_2$ operates over group G_1 and G_2 of prime order $p = \Theta(2^k)$. Global parameters include (e, q, G_1, G_2, g, h, H), where g generates G_1, h is a hash function from an arbitrary string to an element of Z_q, and H is a map-to-point hash function from an arbitrary string to an element of G_1.

- **KeyGen**(1^k): On input of the security parameter 1^k, it generates the public/private key pair (Y_l, x_l) for participant l ($l \in \{A, B_1, B_2, \ldots, B_n, T\}$) where $Y_l = g^{x_l}$ for some random $x_l \in Z_q$.
- **ShareRekey**(x_A, z): On input the delegator's private x_A and the secret value z, this algorithm
  1) generates a warrant message w, which might include the identity information of delegatees and authorized message types;
  2) selects a random polynomial $f(X)$ of degree $t-1$: $f(X) = \sum_{i=0}^{t-1} a_i X_i^i$ such that $f(0) = x_A^{-1}$;
  3) selects random number $\gamma_i \in Z_q$ and computes the pair (R, S_i) such that $R = Y_A^{\beta}$ and $S_i = g^{f(\gamma_i)\beta}$ with $i = 1,2,\ldots,n$ and $\beta = h(w||z)$;
  4) delivers γ_i and (R, S_i) to each delegate B for $i \in \{1,2,\ldots,n\}$ by the secret way.
- **CombineSign**(m, x_{B_i}, σ_{B_i}, γ_i, R, S_i): Before generating a signature, the delegatee B_i requests other delegatees B_j to provide their (R, S_j) and γ_j in a confidential way. A requested delegatee replies if and only if he also planed to sign the message. (Without loss of generality, we assume that the first t delegatees cooperated to sign the message.) After collecting replies from $t-1$ delegatees, the delegatee B_i uses his own private key x_{Bi} and (R, S_i) the delegator provided to compute $\sigma_{B_i} = (\sigma_{B_i}^{(0)}, \sigma_{B_i}^{(1)})$ where

$$\sigma_{B_i}^{(0)} = H(m)^{x_{B_i}} \cdot R;$$

$$\sigma_{B_i}^{(1)} = ((S_i \cdot \prod_{j=1, j\neq i}^{t} S_j)^{\prod_{j=1, j\neq i}^{t} \frac{-\gamma_j}{\gamma_i - \gamma_j}})^{x_{B_i}}.$$

Finally, the delegatee sends the signature and the public key Y_{B_i} to the transformer for resigning.

- **CombineReSign**($\sigma_{B_i}, Y_{B_i}, \beta$): After collecting t different delegatee's signatures and the corresponding public key of delegatees, the transformer first verifies the validity of each signatures received by the following two equations

$$e(\sigma_{B_i}^{(0)}, g) = e(H(m), Y_{B_i}) \cdot e(Y_A, g)^{\beta};$$
$$e(\sigma_{B_i}^{(1)}, Y_A) = e(Y_{B_i}, g)^{\beta}.$$

If both equation hold, the transformer selects a random $\alpha \in Z_q$, and outputs $\sigma_A = (\sigma_A^{(0)}, \sigma_A^{(1)}, \sigma_A^{(2)})$ where

$$\sigma_A^{(0)} = (\prod_{i=1}^{t} (\sigma_{B_i}^{(1)} \cdot Y_A^{-\beta}))^{\alpha};$$
$$\sigma_A^{(1)} = (\prod_{i=1}^{t} Y_{B_i})^{\alpha};$$
$$\sigma_A^{(2)} = (\prod_{i=1}^{t} \sigma_{B_i}^{(1)})^{\beta^{-1}\alpha}.$$

- **Verify**(σ_A, m, Y_A): On input the delegator's public key, the message m, and the signature σ_A, anyone can verify the validity of the signature by the following equations

$$e(\sigma_A^{(0)}, g) = e(H(m), \sigma_A^{(1)});$$
$$e(\sigma_A^{(1)}, g) = e(\sigma_A^{(2)}, Y_A).$$

The algorithm outputs accept if both equations hold, and outputs reject otherwise.

Note that there are two more algorithms in this proposed scheme, but they are not explicitly used in most cases.

- **Registration**(x_A, x_T): The delegator must register with the transformer before he can delegate his signing right to n delegatees. The main objective of this process is to

generate a shared secret z between the delegator and the transformer which allows them to compute the secret value β. This processes is implicit in most cases, for example we can sets $z = Y_A^{x_T} = Y_T^{x_A}$ which is the Diffie-Hellman key of those two participants.

- **ASign**(m, x_A): To sign a message m directly by himself, the delegator selects at random $t \in Z_q$ and outputs $\sigma_{\tilde{A}} = (\sigma_{\tilde{A}}^{(0)}, \sigma_{\tilde{A}}^{(1)}, \sigma_{\tilde{A}}^{(2)})$ where

$$\sigma_{\tilde{A}}^{(0)} = H(m)^{x_A t}\ ;\ \sigma_{\tilde{A}}^{(1)} = Y_A^t\ ;\ \sigma_{\tilde{A}}^{(2)} = g^t.$$

Note that the signature $\sigma_{\tilde{A}}$ can pass the verification algorithm. Moreover, σ_A and $\sigma_{\tilde{A}}$ should be computationally indistinguishable.

B. Analyses

Analyses from the delegator's view: In the proposed scheme, the delegator is capable of delegating his signing right to multiple delegatees in a threshold manner. He uses the proposed scheme to register with the transformer only one time and generates a secret value z. He writes the information of all delegatees (ex. Public key Y_{B_i}) in the warrant message w; thus a delegatee Bi cannot fake a public key to sign signatures on arbitrary messages and ask the transformer to resign. In delegating process, it is very convenient that the delegator does not interact with any delegatee, and no pairing operation is needed. A transformed signature is unforgeable since the following attacks are not possible.

1) A delegate acts as an attacker: The delegatee has (R, S_i) and key pair (Y_{B_i}, x_{B_i}). He wants to forge a transformed signature $\sigma_A' = (\sigma_A^{(0)'}, \sigma_A^{(1)'}, \sigma_A^{(2)'})$ such that

$$\sigma_A^{(0)'} = H(m')^{\alpha' x_{B_i} \cdot \sum_{j=1, j\neq 1}^{t-1} x_{B_j}};$$
$$\sigma_A^{(1)'} = g^{\alpha' x_{B_i} \cdot \sum_{j=1, j\neq 1}^{t-1} x_{B_j}};$$
$$\sigma_A^{(2)'} = g^{\alpha' x_A^{-1} x_{B_i} \cdot \sum_{j=1, j\neq 1}^{t-1} x_{B_j}}.$$

On the one hand, assuming that the delegatee colludes with $t - 1$ delegatees to forge σ_A', the delegatee cannot compute $\sigma_A^{(2)'}$ without the knowledge of β. On the other hand, the delegatee cannot obtain the delegator's private key to compute the value x_A^{-1} from the delegator's public key according the elliptic curve discrete logarithm problem [10]; hence generating $\sigma_A^{(2)'} = (Y_{B_i} \cdot \prod_{j=1}^{t-1} Y_{B_j})^{\alpha' x_A^{-1}}$ by himself is impossible without the value x_A^{-1}.

2) The transformer acts as an attacker: The transformer has verified receiving signature σ_{B_i} and the knowledge of β. He wants to forge a transformed signature $\sigma_A' = (\sigma_A^{(0)'}, \sigma_A^{(1)'}, \sigma_A^{(2)'})$ such that

$$\sigma_A^{(0)'} = H(m')^{\alpha' \sum_{i=1}^{t} x_{B_i}};$$
$$\sigma_A^{(1)'} = (\prod_{i=1}^{t} Y_{B_i})^{\alpha'};$$
$$\sigma_A^{(2)'} = (\prod_{i=1}^{t} \sigma_{B_i}^{(1)})^{\alpha' \beta^{-1}}.$$

In fact, $\sigma_A^{(0)'}$ is basically a Boneh-Lynn-Shacham signature [9], the transformer cannot obtain t delegatees' private key from t delegatees' public key according the elliptic curve discrete logarithm problem [10]; thus generating $\sigma_A^{(0)'}$ by himself is impossible without t delegatees' private key.

3) An outsider acts as an attacker: The outsider has the delegator's public key, and he tries to output a tuple which can pass verification by the Verify algorithm

$$e(\sigma_A^{(0)'}, g) = e(H(m), \sigma_A^{(1)'}); e(\sigma_A^{(1)'}, g) = e(\sigma_A^{(2)'}, Y_A).$$

We assume $\sigma_A' = (\sigma_A^{(0)'}, \sigma_A^{(1)'}, \sigma_A^{(2)'})$ such that

$$\sigma_A^{(0)'} = H(m');\ \ \sigma_A^{(1)'} = g;\ \ \sigma_A^{(2)'} = g^{x_A^{-1}}.$$

The outsider cannot obtain the delegator's private key to compute the value z from the delegator's public key according the elliptic curve discrete logarithm problem [10]; clearly the outsider cannot compute $\sigma_A^{(2)'}$ without the value z.

Analyses from the delegatee's view: In the proposed scheme, an untransformed signature the delegatee signed is not public verifiable. This design protects the privacy of the delegatee. A signature of the delegatee can merely be verified by the delegator and the transformer. Besides, in the delegating process, it is very efficient that any delegatee do not have to interact with the delegator.

From the view of all delegatees, they are all dealer in the proposed scheme. Before signing a signature, every delegatee requests other delegatees to exchange γ_i and S_i in a secret manner. In brief, each delegatee wants to sign a signature σ_{B_i} to the transformer for resigning.

Management is simple in the proposed scheme, the duty of a delegatee is to store and protect the encrypted resigning key. Nobody is capable of obtaining any benefit without the knowledge of β even if the encrypted resigning key is accidentally exposed. For the cost of computation, no pairing operation is needed in the signing process, it cost $t + 1$ multiplication and $t + 1$ exponentiation to generate an untransformed signatures. For the unforgeability of untransformed signatures, using the t out of n technique, it is difficult to forge any delegatee's signature. All following attacks fail to forge an untransformed signature.

1) The delegator acts as an attacker: The delegator has a key pair (Y_A, x_A), the knowledge of β, values (R, γ_i, S_i) for $i \in \{B_1, B_2, \ldots, B_n\}$. He wants to forge an untransformed signature $\sigma_{B_i}' = (\sigma_{B_i}^{(0)'}, \sigma_{B_i}^{(1)'})$ such that

$$\sigma_{B_i}^{(0)'} = H(m')^{x_{B_i}} \cdot R;\quad \sigma_{B_i}^{(1)'} = (Y_{B_i})^{x_A^{-1}\beta}.$$

Actually, $\sigma_{B_i}^{(0)'}$ is basically a Boneh-Lynn-Shacham signature, the delegator cannot obtain any delegatees' private key from any delegatees' public key according the elliptic curve discrete logarithm problem; generating $\sigma_{B_i}^{(0)'}$ by himself is impossible without delegatees' private key.

2) The transformer acts as an attacker: The transformer has delegatee's public key and the knowledge of β. He wants to forge an untransformed signature

$\sigma_{B_i}' = (\sigma_{B_i}^{(0)}', \sigma_{B_i}^{(1)}')$ such that

$$\sigma_{B_i}^{(0)}' = H(m')^{x_{B_i}} \cdot R;$$
$$\sigma_{B_i}^{(1)}' = (Y_{B_i})^{x_A^{-1}\beta}.$$

On the on hand, $\sigma_{B_i}^{(0)}'$ is basically a Boneh-Lynn-Shacham signature [9], it cannot be computed without a delegatee's private key. The transformer cannot obtain any delegatee's private key from any delegatee's public key according to the elliptic curve discrete logarithm problem. On the other hand, the transformer cannot obtain the delegator's private key to compute the value x_A^{-1} from the delegator's public key according to the elliptic curve discrete logarithm problem; hence he cannot compute $\sigma_{B_i}^{(1)}'$ without the value x_A^{-1}.

3) The outsider acts as an attacker: Apparently, without the knowledge of β and private key x_l of $l \in \{A, B_1, B_2, \ldots, B_n\}$, a malicious outsider cannot forges an untransformed signature $\sigma_{B_i}' = (\sigma_{B_i}^{(0)}', \sigma_{B_i}^{(1)}')$ such that

$$\sigma_{B_i}^{(0)}' = H(m')^{x_{B_i}} \cdot R;$$
$$\sigma_{B_i}^{(1)}' = (Y_{B_i})^{x_A^{-1}\beta}.$$

Besides unforgeability, an untransformed signature is anonymous, hence any delegatee has not to worry about the possibility that a malicious person forges an untransformed signature to injure his benefits.

Analyses from the transformer's view: In comparison with [3]-[5], the transformer does not have to save and protect a resigning key, the proposed scheme reduces the cost of store and preserve resigning key, he only stores and preserves the secret value *z*. Besides, it is very efficient that the transformer registers is non-interactive in delegation to multiple delegates.

In the proposed scheme, the transformer is in fact a clerk that control and manage receiving at least *t* delegatees' signature. He verifies signatures of delegatees and combines all receiving signature, he can reject resigning if this check fails.

C. Possible Application

In this paper, it is our major objective that designing a proxy re-signature scheme allows delegation of signing right to multiple delegatees. Delegating signing right to multiple delegatees securely is the most straightforward application of the proposed scheme. A manager (The delegator *A*) may delegates his signing right to his secretary (The transformer *T*) and multiple employees (delegatees B_i). If a manager goes on a business trip, he assigns his secretary and multiple employees to deal emergent document such as a contract, etc. Firstly, the manager broadcasts a warrant message *w* and multiple employees' public key, he allows multiple employees to sign a signature σ_A on his behalf if the secretary cooperated. The manager generates γ_i and (R, S_i) with a random polynomial $f(X)$, then he sends it to the multiple employees in a secure way, respectively. Before signing a signature, it is necessary for an employee to exchange his γ_i and S_i with at least $t-1$ employees. Then they send respective signatures to the secretary. After receiving multiple signatures, the secretary combines all multiple signatures and translates into the signature σ_A, if multiple signatures pass all necessary verifications respectively. In this way, the manager can concentrates on the business trip, he does not worry about the possibility that an emergent document is neglected. The proposed scheme not only allows delegation of signing right to multiple employees but also strengthens the security of all participants.

Please note that the transformer must collect at least *t* delegatees' signatures to combine and translate into the signature of the delegator, this scenario is in some sense not flexible.

In fact, there is an alternative usage of the proposed scheme. A delegatee's signature shows *t* delegatees agreed with signing the same message *m*, it already satisfies *t* out of *n* threshold since the transformer receives a delegatee's signature and pass verification. Therefore, the transformer has not to receive at least other $t-1$ delegatees' signatures to combine and translate into the signature of the delegator, he can translates a delegatee's signature into the delegator's signature directly.

For example, in an anonymously signing document, the delegatee collects other $t-1$ delegatees' γ_i and S_i to sign a message and deliver his signature to the transformer. The transformer can transform the delegaees' signatures into the delegator's signature directly. In this way, the proposed scheme becomes more convenient and efficient.

IV. Conclusion

In this paper, we propose a unidirectional single-use threshold proxy re-signature scheme with privacy. The proposed scheme considers the privacy of the delegatee and can be applied in the scenario involving multiple delegatees.

References

[1] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signature for delegating signing operation," *ACM Conference on Computer and Communications Security*, 1996, pp. 48-57.

[2] S. Kim, S. Park, and D. Won, "Proxy signature, revisited," *ICICS*, 1997, pp. 223-232

[3] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and automatic proxy cryptography," *EUROCRYPT*, 1998, pp. 127-144.

[4] G. Ateniese and S. Hohenberger, "Proxy re-signatures: new definitions, algorithms, and applications," *ACM Conference on Computer and Communications Security*, 2005.

[5] B. Libert and D. Vergnaud, "Multi-use unidirectional proxy re-signatures," *ACM Conference on Computer and Communications Security*, 2008, pp. 511-520.

[6] G. Y. Chen and H. C. Lin, "Proxy re-signatures with warrant and privacy considerations," *Cryptology and Information Security Conference*, Taiwan, 2012.

[7] A. Shamir, "How to share a secret," *Commun. ACM 22*, pp. 612-613, 1979.

[8] Y. Li, H. Lipmaa, and D. Pei, "On delegatability of four designated verifier signature," *ICICS*, 2005, pp. 61-71.

[9] D. Boneh, B. Lynn, and H. Shacham, "Short signature from the weil pairing," *ASIACRYPT*, 2001, pp. 514-532.

[10] N. Koblitz, "Elliptic cure cryptosystems," in *Mathematic of Computation*, vol. 48, pp. 203-209, 1987.

Kuan-Yu Chen received the B.S. degree in computer science and information engineering from Aletheia University, Taiwan, R.O.C. in 2010. He is a graduate student of Aletheia University since September 2010. His research interest includes cryptography and digital signature.

Hsi-Chung Lin received his Ph.D. degree in computer science and information engineering from the National Central University, Taiwan, in 2008. He is an assistant professor of the Dept. of Computer Science and Information Engineering, Aletheia University, Taiwan, R.O.C. His research interests include the designs, the applications, and analysis of various cryptographic primitives.