J. Software Engineering & Applications, 2009, 2: 267-275
doi:10.4236/jsea.2009.24034 Published Online November 2009 (http://www.SciRP.org/journal/jsea)

Secure Chained Threshold Proxy Signature without and with Supervision*

Zoe L. JIANG, S. M. YIU, Y. DONG, L. C. K. HUI, S. H. Y. WONG

Department of Computer Science, The University of Hong Kong, Hong Kong, China.
Email: {ljiang, smyiu, ydong, hui, shywong}@cs.hku.hk.

Received June 4th, 2009; revised July 15th, 2009; accepted July 24th, 2009.

ABSTRACT

Threshold Proxy Signature (TPS) scheme facilitates a manager to delegate his signing capability to a group of n_2 subordinates without revealing his own private key, such that a subgroup of at least $t_2 \leqslant n_2$ subordinates is required to generate a proxy signature. In reality, the situation can be more complicated. First of all, the subgroup may further delegate their proxy signing capabilities to another group of n_3 subordinates such that at least another subgroup of at least $t_3 \leq n_3$ subordinates are of the proxy signing capabilities (in the form of a chain). t_2 can be unequal to t_3 depending on the concrete requirement. This is a group-to-group delegation problem. In addition, a supervising agent (SA) may be introduced in the above chain to supervise the subordinates, such that proxy signing can only be successfully executed with SA's agreement. This is a delegation with supervision problem in the threshold delegation chain described above. These two extensions of delegation problems are not solved yet. This paper designs two provably secure cryptographic schemes Chained Threshold Proxy Signature (CTPS) scheme and Chained Threshold Proxy Signature with Supervision (CTPSwS) scheme to solve these two delegation problems.

***Keywords*:** *Delegation, Threshold Proxy Signature, Chained Threshold Proxy Signature with Supervision*

1. Introduction

1.1 Motivation

It is a common practice for a manager to delegate his signing right to a group of n subordinates when he is on leave so that a subgroup of at least $t \leq n$ of them can cooperate to sign a document on behalf of the manager. This is a threshold delegation problem, and can be solved by Threshold Proxy Signature (TPS) [2] scheme. In reality, the delegation may involve more than one level (in the form of a chain). Consider the following scenario. There is an email sent by the manager of software development department in corporation A, Simon, to the manager of the same department in corporation B, Sam. Since Sam is too busy to check every single detail of the data part before he replies with his signature, and the data are so important that he cannot rely on any single one of his three vice managers (his subordinates), he forwards this email to all of them. Further any two of them, as a subgroup, may forward this email to their employees checking the data part. As a result, to answer this email to Simon, it is desirable for a subgroup of the employees to cooperate on behalf of Sam. How the any two of the vice managers pass their proxy signing capabilities to their employees is referred to group-to-group delegation problem. In a more cautious situation, the contract part needs to be authorized by the manager of the software maintenance department in corporation B, Steven. In this case, besides delegation, Sam is also required to appoint Steven as his supervising agent (SA) such that only when Steven agrees to the contract part, the employees can compute a proxy signature on behalf of Sam. How Sam appoints Steven as an SA is referred to delegation with supervision problem in threshold delegation chain. In this paper, we propose two schemes, Chained Threshold Proxy Signature (CTPS) scheme and Chained Threshold Proxy Signature with Supervision (CTPSwS) scheme, to solve these two delegation problems.

1.2 Related work

Mambo *et al*. [3] introduced the first efficient proxy signature in 1996, where it allows a user to delegate his signing power to a designated signer, a proxy signer. It is widely applicable in all kinds of known standard signature schemes such as El Gamal scheme [4], Okamoto scheme [5] and Fiat-Shamir scheme [6]. In 1997, Kim *et al*. [2] proposed proxy signature for partial delegation with warrant combining the benefit of Mambo's partial delegation and Neuman's [7] delegation by warrant/certi-

*The preliminary work has been published in 2008 International Conference on Computer Science and Software Engineering [1].

Copyright © 2009 SciRes

ficate. They also extended it to a (t, n) Threshold Proxy Signature (TPS) such that any $t \leq n$ proxy signers using their proxy secret key shares can cooperate to generate a proxy signature on behalf of an original signer, but less than t can not, by deploying Ceredo's Schnorr type threshold digital signature scheme [8].

Lui *et al*. [9] proposed a chained delegation scheme with supervision scheme, in which the original signer sends, in ahead of time, his permission information about the proxy signer to a supervising agent (SA) who he trusts. Then the proxy signer can only generate a valid proxy signature under SA's supervision even when the original signer is not available. This delegation can be executed in multiple levels. However, on one hand, the scheme does not consider the threshold problem. On the other hand, the scheme sacrificed both the original signer and the supervising agent's undeniabilities due to the advantage the authors presented that there is no need for the verifier to be aware of whether supervision is performed or not. In many cases, this is unacceptable from the security point of view.

Boldyreva [10] defined a formal proof model for the security of proxy signature schemes, which enables the cryptographic analysis of such schemes, instead of just presenting attacks that fail. Then they proved the security of Triple Schnorr Proxy Signature scheme, a variant of Kim at al.'s proxy signature scheme, preserving its efficiency, in the random oracle model assuming the hardness of computation of discrete logarithm.

1.3 Our Contribution

Firstly, in this paper we propose Chained Threshold Proxy Signature (CTPS) scheme to solve the group-to-group delegation problem. Although, Threshold Proxy Signature (TPS) scheme [2] and Triple Schnorr Signature scheme [10] are two important components to design our scheme, we need to consider how to distribute the proxy shares from a subgroup of vice managers to another subgroup of employees in a group-to-group manner. Therefore, we deploy Herzberg *et al*.'s [11] proactive secret sharing idea into our scheme. Proactive secret sharing is proposed to periodically renew the shares without changing the secret, in such a way that any information learnt by the adversary about individual shares becomes obsolete after the shares are renewed. But in our scheme, the renewed shares should be securely passed to employees by each vice manager while old ones are kept secret by the vice managers themselves.

To solve the delegation with supervision problem in threshold delegation chain, we adapt Lui *et al*'s supervision idea in delegation chain (no threshold) into CTPS scheme to implement Chained Threshold Proxy Signature with Supervision (CTPSwS) scheme. Different with Lui *et al*'s idea, however, supervising agent is also actively involved in the delegation using his/her own private key, such that verification for the proxy signature requires supervising agent's public key as well, besides original signer and proxy signers' public keys.

We also provide formal security models and proofs to show that the schemes we designed are secure in the random oracle model assuming the hard problem of discrete logarithm.

1.4 Organization

A 3-level CTPS scheme and its security proof will be described in section 2 & 3, respectively. Then a 3-level CTPSwS scheme and security proof draft will be introduced in section 4 & 5, respectively. Section 6 concludes the paper and discusses some future work.

2. Chained Threshold Proxy Signature (CTPS)

Recall that Sam, the manager of software development department in corporation B, delegates his signing right to a group of n_2 vice managers, any subgroup of whom with t_2 members further delegate their proxy signing capabilities to a group of n_3 employees, such that any subgroup of t_3 employees can reply to Simon with their proxy signature on behalf of Sam.

We can formally define the above roles by letting Sam the original signer u_1 in level 1, vice managers a group of n_2 proxy signers in level 2, ($\{u_{2,j}\}_{n_2}$ for short), and employees a group of n_3 proxy signers in level 3, ($\{u_{3,k}\}_{n_3}$ for short). Any subgroup of $t_2 \leq n_2$ vice managers performing the delegation is defined as U_2. Similarly, any subgroup of $t_3 \leq n_3$ employees signing the replied email is defined as U_3. Note the difference between $\{u_{2,j}\}_{n_2}$ and U_2, $\{u_{3,k}\}_{n_3}$ and U_3. WLOG, we assume $U_2 = \{u_{2,1}, u_{2,2}, \cdots, u_{2,t_2}\} = \{u_{2,j}\}_{t_2}$ and $U_3 = \{u_{3,1}, u_{3,2}, \cdots, u_{3,t_3}\} = \{u_{3,k}\}_{t_3}$. Let ($sk_1$,$pk_1$), ($sk_{2,j}$, $pk_{2,j}$), (skg_2, pkg_2), ($sk_{3,k}$, $pk_{3,k}$) and (skg_3, pkg_3) denote u_1, $u_{2,j}$, $\{u_{2,j}\}_{n_2}$, $u_{3,k}$, and $\{u_{3,k}\}_{n_3}$'s secret/public key pairs respectively. By extending Boldyreva's proxy signature scheme model, CTPS scheme to achieve the delegation procedure should involve a one-to-group protocol run between the original signer and the group of proxy signers in level 2, a group-to-group protocol run between any subgroup of proxy signers in level 2 and the group of proxy signers in level 3, a chained threshold proxy signing algorithm and the corresponding verification algorithm. Additionally, there should be an algorithm that extracts the identities of the groups of proxy signers in both level 2 & 3.

Definition 1 describes the detailed components of a 3-level Chained Threshold Proxy Signature scheme. A list of important parameters and symbols is shown here for your reference.

ω_1: warrant including u_1 and $\{u_{2,j}\}_{n_2}$'s IDs, and other

Copyright © 2009 SciRes

information on the delegation

ω_2: warrant including $\{u_{2,j}\}_{n_2}$ and $\{u_{3,k}\}_{n_3}$'s IDs, and other information on the delegation

skt: secret key transformation generated by u_1

$skt_{1,j}$: share of secret key transformation sent to $u_{2,j}$

$skp_{2,j}$: proxy secret key share generated by $u_{2,j}$

$skt_{2,j}$: share of proxy secret key transformation generated by $u_{2,j}$

$skt_{2,j,k}$: sub-share of proxy secret key transformation sent to $u_{3,k}$

$skt'_{2,k}$: share of proxy secret key transformation retrieved by $u_{3,k}$

$skp_{3,k}$: proxy secret key share generated by $u_{3,k}$

$p\sigma_3$: chained threshold proxy signature.

Definition 1 (CTPS Scheme) *Let* ***CTPS*** = (*G, K, (TD, TP), (CTD, CTP), CTPS, CTPV, CTPID*) *be a chained threshold proxy signature scheme, where the constituent algorithms run in polynomial time*.

G is a random *parameter-generation* algorithm, and it will output some global parameters params.

K is a random *key-generation* algorithm, and it will output secret/public key pairs for original signer u_1 and proxy signers in both level 2 & 3, $\{u_{2,j}\}_{n_2}$ and $\{u_{3,k}\}_{n_3}$, in the scheme.

(*TD, TP*) is a *Threshold Designation-Proxy* protocol between the original signer u_1 and the proxy signers in level 2, $\{u_{2,j}\}_{n_2}$. Both *TD* and *TP* take as input the public keys pk_1 and pkg_2, respectively. *TD* also takes as input the secret key sk_1 of u_1, and *TP* also takes as input the secret key $sk_{2,j}$ of $u_{2,j}$. As the result of the interaction, the expected local output of *TP* is $skp_{2,j}$, the proxy secret share which is kept secret by each $u_{2,j}$.

$$[TD\ (pk_1, pkg_2, sk_1), TP\ (pk_1, pkg_2, sk_{2,j})] \rightarrow skp_{2,j}$$

(*CTD, CTP*) is a *Chained Threshold Designation-Proxy* protocol between U_2 and $\{u_{3,k}\}_{n_3}$. Both *CTD* and *CTP* take as input the public keys pk_1, pkg_2 and pkg_3, respectively. *CTD* also takes as input the proxy secret shares $\{skp_{2,j}\}_{t_2}$ of $\{u_{2,j}\}_{t_2}$. *CTP* takes as input the secret key $sk_{3,k}$ of $u_{3,k}$. The expected local output of *CTP* is $skp_{3,k}$, the proxy secret key share which is kept secret by each $u_{3,k}$. Note that for each $u_{3,k}$ to generate $skp_{3,k}$, the subgroup U_2 is involved in *CTD*, but not just a certain proxy signer in U_2.

$$[CTD\ (pk_1, pkg_2, pkg_3, \{skp_{2,j}\}_{t_2}),$$
$$CTP\ (pk_1, pkg_2, pkg_3, sk_{3,k})] \rightarrow skp_{3,k}$$

CTPS is the (possibly) randomized *Chained Threshold Proxy Signing* algorithm. It takes as input $\{skp_{3,k}\}_{t_3}$ and a message $M \in \{0,1\}^*$, and outputs a chained threshold-proxy signature $p\sigma_3$.

$$CTPS\ (M, \{skp_{3,k}\}_{t_3}) \rightarrow p\sigma_3$$

CTPV is the deterministic *Chained Threshold Proxy Verification* algorithm. It takes as input a message M, a proxy signature $p\sigma_3$, and (pk_1, pkg_2, pkg_3), and outputs 0 or 1.

$$CTPV\ (M, p\sigma_3, pk_1, pkg_2, pkg_3) = 0/1$$

CTPID is the *Chained Threshold Proxy IDentification* algorithm. It takes as input a valid proxy signature $p\sigma_3$ and outputs identities of two proxy signer groups, i.e., public keys.

$$CTPID\ (p\sigma_3) = (pkg_2, pkg_3)/\perp$$

SIGNATURE VERIFICATION CONDITION: If $CTPV = 1$ and $CTPID = (pkg_2, pkg_3)$, we say $p\sigma_3$ is a valid chained threshold proxy signature by proxy signers in U_2 and U_3 on behalf of u_1.

The definition clearly describes what kinds of individual algorithms and interactive protocols are required to be run by original signer and proxy signers. After define the structure of CTPS scheme, we design a concrete scheme based on the definition.

We give a high-level description of our scheme here, followed by the concrete calculation. First of all, by using public parameters and secret/public key pairs generated through G and K, u_1 generates the certificate of warrant ω_1 in (1), which is actually a signature of ω_1 using sk_1. We call it secret key transformation skt_1 in our scheme since it masks u_1's secret key sk_1 and will be used for $\{u_{2,j}\}_{n_2}$ to generate proxy signing keys. In order to designate $\{u_{2,j}\}_{n_2}$ as u_1's threshold proxy signers, each share $skt_{1,j}$ generated by (2) will be distributed to $u_{2,j}$ securely. After verifying $skt_{1,j}$ as a signature generated by u_1 using (3), each $u_{2,j}$ computes proxy secret key share $skp_{2,j}$ in (4).

As the applications we described above, if any $t_2 \le n_2$ vice managers, such as U_2, want to further delegate their proxy signing capabilities to their employees $\{u_{3,k}\}_{n_3}$, which we call a group-to-group delegation, each $u_{2,j} \in U_2$ computes secret key transformation $skt_{2,j}$ in (5) as u_1 does in (1), then divides it to n_3 shares, $skt_{2,j,k}$(k=1,2, ⋯ ,n_3), as calculated in (6), which are sent to $u_{3,k}$ securely. As a result, each $u_{3,k}$ verifies $skt_{2,j,k}$(k=1,2, ⋯ ,t_2) he receives using (8) and computes $skt'_{2,k}$ by accumulating them in (9). By comparing (5) and (9), $skt_{2,j}$ and $skt'_{2,k}$ are generated by two different random polynomials $F_2(j)$ and $F'_2(k)$ with same constant $sktg_2$. For how (9) is deduced, please refer to Lagrange Formula, which was also used in [2]. Then each $u_{3,k}$ can successfully generate the proxy secret key share $skp_{3,k}$ in (11).

Copyright © 2009 SciRes

Let us discuss a little more about the difference and difficulty of (*CTD, CTP*) protocol compared with (*TD, TP*) here. During (*TD, TP*) protocol, $skt_{1,j}$ carrying secret information sk_1 inside is generated and delivered to $u_{2,j}$ as the mark for u_1 to designate $u_{2,j}$ as one of his proxy signers. Similarly in (*CTD, CTP*) protocol, $skt_{2,j}$ carrying secret information $sk_{2,j}$ should also be delivered to $\{u_{3,k}\}_{n_3}$, but in an indirect way for the reason that there are a group of delegators and a group of delegatees. Sending $skt_{2,j}$ to $u_{3,k}$ where $j = k$ one by one does not work because U_2 and $\{u_{3,k}\}_{n_3}$ may have different numbers t_2 and n_2. However, from the group point of view, we need a scheme to reshuffle $\{skt_{2,j} = F_2(j)(j = 1,2, \cdots, t_2)\}$ to $\{skt'_{2,k} = F'_2(k)(k = 1, 2, \cdots, n_3)\}$, satisfying that $F_2(0) = F'_2(0) = sktg_2$. It seems that we keep the group secret key transformation $sktg_2$ unchanged and make secret key transformation shares updated. With this purpose, we found a good candidate of proactive secret sharing approach [11], which was proposed to periodically renew the shares, like $\{skt_{2,j}\}_{t_2}$, to the new ones, like $\{skt'_{2,k}\}_{n_3}$, without changing the secret, like $sktg_2$. However, in our scheme, the renewed ones belong to $\{u_{3,k}\}_{n_3}$, but not $\{u_{2,j}\}_{n_2}$.

Schnorr signature scheme [12] is used in *CTPS* and *CTPV* where partial proxy signatures $\{ps_{3,k}\}_{t_3}$ are published among U_3 such that each $u_{3,k} \in U_3$ can calculate proxy signature $p\sigma_3$ in (13). The following details how the algorithms and protocols are performed.

The system runs *G*. On input 1^k, params = $G(1^k)$ = (p,q,g,G,H), such that $2^{k-1} \le p < 2^k$, $q|p-1$, $g \in Z_p^*$ of order q, two hash functions G: $\{0,1\}^* \to Z_q$, and H : $\{0,1\}^* \to Z_q$.

The system runs *K*. On input (p,q,g,G,H), *K* generates $sk_1 \in_R Z_q$, $pk_1 = g^{sk_1} \bmod p$, $sk_{2,j} = SK_2(j)$ and $pk_{2,j} = g^{sk_{2,j}} \bmod p$, where $SK_2(x)$ is a random polynomial with random constant skg_2 and degree $t_2 - 1$. $pkg_2 = g^{skg_2} \bmod p$. Note that although the intuitive idea for the above procedure is to generate $SK_2(x)$ and distribute $sk_{2,j}$ to $u_{2,j}$ securely by a trusted dealer, we deploy the protocol for generating random number in [2] to implement it without a dealer for security consideration. As a result, each $u_{2,j}$ can only calculate his own secret key $sk_{2,j}$ without knowing skg_2. skg_3, pkg_3, $\{sk_{3,k}\}_{n_3}$ and $\{pk_{3,k}\}_{n_3}$ are generated in the same way.

u_1 runs *TD*.

$$skt_1 = e_1 \cdot sk_1 + k_1 \bmod q, \text{ where} \quad (1)$$

$$r_1 = g^{k_1} \bmod p, k_1 \in_R Z_q^*,$$

$$e_1 = G(0 \| pk_1 \| pkg_2 \| \omega_1, r_1).$$

$$skt_{1,j} = F_1(j) = skt_1 + a_{1,1}j + a_{1,2}j^2 + \cdots + a_{1,t_2-1}j^{t_2-1} \bmod q. \quad (2)$$

$F_1(j)$ is a random polynomial privately owned by u_1. r_1 and $\{g^{a_{1,m}} \bmod p\}\ t_2 - 1$ are broadcast.

$u_{2,j}$ runs *TP*.

$$g^{skt_{1,j}} = (pk_1^{e_1} r_1) \cdot A \bmod p, \text{ where} \quad (3)$$

$$A = \prod_{m=1}^{t_2-1} (g^{a_{1,m}})^{j^m} \bmod p.$$

$$skp_{2,j} = e_1 \cdot sk_{2,j} + skt_{1,j} \bmod q. \quad (4)$$

$u_{2,j}$ runs *CTD*.

$$\begin{aligned} skt_{2,j} &= e_2 \cdot skp_{2,j} + k_{2,j} \bmod q \\ &= e_2(e_1 \cdot SK_2(j) + F_1(j)) + K_2(j) \\ &= F_2(j), \text{ where} \end{aligned} \quad (5)$$

$$e_2 = G(0 \| pkg_2 \| pkg_3 \| \omega_2, r_2),$$

$$F_2(0) = e_2(e_1 \cdot skg_2 + skt_1) + k_2 = sktg_2.$$

$$skt_{2,j,k} = F_{2,j}(k) \bmod q, \text{ where} \quad (6)$$

$$F_{2,j}(k) = skt_{2,j} + b_{2,j,1}k + b_{2,j,2}k^2 + \cdots + b_{2,j,t_3-1}k^{t_3-1} \bmod q. \quad (7)$$

$\{k_{2,j}\}_{n_2}$ are generated by running the protocol for generating random number in [2] such that each $u_{2,j}$ can calculate $k_{2,j}$ without knowing any other's secret shares $\{k_{2,x}\}_{x \ne j}$ and satisfying that $k_{2,j} = K_2(j)$, where $K_2(j)$ is a random polynomial with constant kg_2 and degree $t_2 - 1$. $F_{2,j}(k)$ is a random polynomial privately owned by each $u_{2,j}$ with constant $skt_{2,j}$ and degree $t_3 - 1$. $\{r_{2,j} = g^{k_{2,j}} \bmod p\}\ t_2$ and $r_2 = g^{kg_2} \bmod p$ are broadcast.

$u_{3,k}$ runs *CTP*.

$$g^{skt_{2,j,k}} = \left(\left(pk_1 pk_{2,j}\right)^{e_1} r_1 A\right)^{e_2} r_{2,j} B, \text{ where} \quad (8)$$

$$B = \prod_{m=1}^{t_3-1} (g^{b_{2,j,m}})^{k^m} \bmod p$$

$$\begin{aligned} skt'_{2,k} &= \sum_{j=1}^{t_2} \delta_j skt_{2,j,k} = \sum_{j=1}^{t_2} \delta_j F_{2,j}(k) \\ &= F'_2(k), \text{where} \end{aligned} \quad (9)$$

$$F'_2(0) = \sum_{j=1}^{t_2} \delta_j F_{2,j}(0) = \sum_{j=1}^{t_2} \delta_j skt_{2,j} = sktg_2 \quad (10)$$

$$\delta_j = \prod_{l=1, l \ne j}^{t_2} -l / j - l$$

$$skp_{3,k} = e_2 \cdot sk_{3,k} + skt'_{2,k} \bmod q. \quad (11)$$

U_3 runs *CTPS*.

$$ps_{3,k} = c_3 \cdot skp_{3,k} + k_{3,k}, \text{ where} \quad (12)$$

$$c_3 = H(1 \| M \| pkg_2 \| pkg_3 \| \omega_2 \| r_1, r_2), \text{ and}$$

$$r_3 = g^{k_3} \bmod p.$$

$$p\sigma_3 = \sum_{u_{3,k} \in U_3} \delta_k \cdot ps_{3,k}, \quad (13)$$

Copyright © 2009 SciRes

$$\delta_k = \prod_{l=1,l\neq k}^{t_3} -l \big/ k-l$$

Verifier runs *CTPV*.

$$g^{p\sigma_3} = PKP^{c_3} \cdot r_3 \bmod p, \text{ where} \quad (14)$$

$$PKP = \left(pk_1 \cdot pkg_2\right)^{e_1} \cdot r_1^{e_2} \cdot r_2 \cdot pkg_3^{e_2}.$$

PROOF OF (14) Let θ_3 represents $u_{3,k} \in U_3$.

$$\begin{aligned}
p\sigma_3 &= \sum_{\theta_3} \delta_k ps_{3,k} = \sum_{\theta_3} \delta_k (c_3 + skp_{3,k} + k_{3,k}) \\
&= c_3 \sum_{\theta_3} \delta_k skp_{3,k} = \sum_{\theta_3} \delta_k k_{3,k} \\
&= c_3 \sum_{\theta_3} \delta_k (e_2 sk_{3,k} + skt'_{2,k}) + k_3 \\
&= c_3 (e_2 \sum_{\theta_3} \delta_3 sk_{3,k} + \sum_{\theta_3} \delta_k skt'_{2,k}) + k_3 \\
&= c_3 (e_2 skg_3 + \sum_{\theta 2} \delta_j skt_{2,j}) + k_3 \\
&= c_3 (e_2 skg_3 + \sum_{\theta 2} \delta_j (e_2 skp_{2,j} + k_{2,j})) + k_3 \\
&= c_3 (e_2 skg_3 + e_2 \sum_{\theta 2} \delta_j skp_{2,j} + k_2) + k_3 \\
&= c_3 (e_2 skg_3 + e_2 (e_1 (skg_2 + sk_1) + k_1) + k_2) + k_3 \bmod q, \\
L.H.S. &= g^{p\sigma_3} \\
&= g^{c_3 (e_2 skg_3 + e_2 (e_1 (skg_2 + sk_1) + k_1) + k_2) + k_3} \bmod p \\
&= R.H.S.
\end{aligned}$$

The proof proves the correctness of our CTPS scheme from the computation point of view. In the following section, we will prove its security.

3. Security of the CTPS Scheme

In the section, we set up CTPS security model and prove that our CTPS scheme is secure against adaptive chosen-message attack in random oracle model.

As discussed in [10], the formal model includes a rather powerful adversary who is able to corrupt all other users' secret keys except the *Single Honest User* (*SHU*). Furthermore, A is of the capability to launch adaptive chosen-message attacks according to three kinds of roles that the *SHU* can play, namely *Role 1*: the original signer u_1, *Role 2*: one of the required proxy signers in U_2, say $u_{2,2}$, and *Role 3*: one of the required proxy signers in U_3, say $u_{3,2}$. All kinds of attacks will be described in the model later. Besides, A can access to a chained threshold proxy signing oracle. So the goal of the adversary A includes:

- A forgery *CTPS* on behalf of u_1 (*SHU*);
- A forgery *CTPS* by proxy signers in U_3, who are delegated by U_2 including $u_{2,2}$ (*SHU*), on behalf of u_1;
- A forgery *CTPS* by proxy signers including $u_{3,2}$ (*SHU*) in U_3 on behalf of u_1.

Definition 2 (Security of CTPS Scheme) *Let* ***CTPS*** = (*G*, *K*, (*TD*, *TP*), (*CTD*, *CTP*), *CTPS*, *CTPV*, *CTPID*) *be a chained threshold proxy signature scheme. Consider an experiment* $\boldsymbol{Exp}_{CTPS,A}(k)$ *related to* ***CTPS****, adversary A and parameter k. In the extreme case, adversary A should represent all proxy signers if the SHU is the original signer; or the original signer and all other proxy signers except the SHU if the SHU is one of the proxy signers in level 2 or 3. First, system parameters* params *and secret/public key pairs are generated by running G and K. Empty array* $\boldsymbol{skp}_{3,2}$ *and empty sets* $\boldsymbol{pkg}_2$ *and* $\boldsymbol{pkg}_3$ *are created. The adversary A is of all the secret keys except SHU's, and it can make the following requests and queries.*

R1: u_1(*SHU*) designates $\{u_{2,j}\}_{n_2}$ and $\{u_{3,k}\}_{n_3}$ as his proxy signers. *A* requests to interact with u_1(*SHU*) running *TD*, and plays the role of $\{u_{2,j}\}_{n_2}$ running *TP*, and the roles of U_2 and $\{u_{3,k}\}_{n_3}$ running (*CTD*, *CTP*). And $\boldsymbol{pkg}_2$ is set to $\boldsymbol{pkg}_2 \cup pkg_2$.

R2: u_1 designates $\{u_{2,j}\}_{n_2}$ and $\{u_{3,k}\}_{n_3}$, where $u_{2,2}$ is the *SHU*, as his proxy signers in level 2 and 3 respectively. *A* requests to interact with $\{u_{2,j}\}_{n_2}$ running *TP*, and plays the role of u_1 running *TD* and the roles of all other proxy signers in level 2 except $u_{2,j}$. Then *A* requests again to interact with $u_{2,2}$ running *CTD*,, and plays the role of U_2 except $u_{2,2}$, and $\{u_{3,k}\}_{n_3}$ running (*CTD*, *CTP*). $\boldsymbol{pkg}_3$ is set to $\boldsymbol{pkg}_3 \cup pkg_3$.

R3: u_1 designates $\{u_{2,j}\}_{n_2}$ and $\{u_{3,k}\}_{n_3}$, where $u_{3,2}$ is the *SHU*, as his proxy signers in level 2 and 3 respectively. *A* requests to interact with $\{u_{3,k}\}_{n_3}$ running *CTP*, and plays the role of U_2 running *CTD*, and the roles of u_1 and $\{u_{2,j}\}_{n_2}$ running (*TD*, *TP*). The private output $\boldsymbol{skp}_{3,2}$ by $u_{3,2}$ (*SHU*) is stored in $\boldsymbol{skp}_{3,2}$. *A* does not have access to $\boldsymbol{skp}_{3,2}$.

Q1: Chained threshold proxy signature query by U_3, where $u_{3,2}$ is the *SHU*, on behalf of u_1. *A* can make a query (*M*, 32, *x*) to oracle O_{CTPS} ($skp_{3,k}$(k = 1,3, ⋯ ,t_3), $\boldsymbol{skp}_{3,2}[x]$,·,·,·,). If $\boldsymbol{skp}_{3,2}[x]$ has been defined, we say that this query is valid and the oracle returns $p\sigma_3 = O_{CTPS}$ ($skp_{3,k}$(k = 1,3, ⋯ ,t_3), $\boldsymbol{skp}_{3,2}[x]$,*M*,32,*x*,). Eventually *A* outputs a forgery (*M*, $p\sigma_3$, pk_1). The output of the experiment is 1, if

- *CTPID* ($p\sigma_3$) \ $pkg_3 \notin \boldsymbol{pkg}_2$, or
- *CTPID* ($p\sigma_3$) \ $pkg_2 \notin \boldsymbol{pkg}_3$, or
- No valid query (*M*, 32, *x*)to O_{CTPS} ($skp_{3,k}$(k = 1,3, ⋯ ,t_3), $\boldsymbol{skp}_{3,2}[x]$,·,·,·,).

Otherwise, the output is 0.

We define the advantage of adversary A as

$\boldsymbol{Adv}_{CTPS,A}(k) = \Pr[\boldsymbol{Exp}_{CTPS,A}(k) = 1]$.

We say that ***CTPS*** is a secure chained threshold proxy signature scheme if the function $\boldsymbol{Adv}_{CTPS,A}(k)$ is negligible for *A* of time complexity polynomial in the security parameter *k*.

SECURITY OF CTPS. The following theorem states

Copyright © 2009 SciRes

our result about the security of Chained Threshold Proxy Signature scheme. The proof of Theorem 1 is in Appendix A.

Theorem 1 *Let* ***CTPS*** = (*G, K,* (*TD, TP*), (*CTD, CTP*), *CTPS, CTPV, CTPID*) *be our proposed chained threshold proxy signature scheme in random oracle model. If the Schnorr signature scheme is secure, then* ***CTPS*** *scheme is secure in random oracle model.*

PROOF IDEA. The conclusion that a Chained Threshold Proxy Signature scheme is provably secure can be deduced with respect to the contradiction that if a forgery of a chained threshold proxy signature scheme by *A* successes in polynomial time, i.e. $\boldsymbol{Adv}_{CTPS,A}(k)$ is not negligible, then a well-known standard signature scheme, i.e. the Schnorr signature scheme, is broken.

4. Chained Threshold Proxy Signature with Supervision (CTPSwS) Scheme

Recall when Sam, the software development department, appoints Steven, the software maintenance department, as the supervising agent to supervise proxy signers such that only with permission of Steven, proxy signers can perform the signing capabilities. *CTPSwS* scheme in this section extended from *CTPS* fits into this kind of scenario by deploying Lui *et al.*'s [9] idea into our *CTPS* scheme.

Different from the *CTPS* model, there is a new protocol (TD_{SA}, TP_{SA}) run between original signer u_1 and his supervising agent SA_1. To differentiate the protocol run between u_1 and $\{u_{2,j}\}_{n_2}$ in *CTPSwS* scheme with that in *CTPS*, we define the former as (TD_u, TP_u). The signing and verification algorithm should also include SA_1's public key. The *CTPSwS* scheme model is as follows in Definition 3. Besides the notations described in Section 2, we have several new ones shown here.

(sk_{SA_1}, pk_{SA_1}) : SA_1's secret and public key pair

skp_{SA_1} : partial proxy secret key generated by SA_1

ps_{SA_1} : partial chained threshold proxy signature generated by SA_1

Definition 3 (CTPSwS Scheme) *Let* ***CTPSwS*** = {*G, K,* (TD_{SA},TP_{SA}), (TD_u,TP_u), (*CTD,CTP*), *CTPSwS, CTPVwS, CTPIDwS*} *be a chained threshold proxy signature with supervision scheme, where the constituent algorithms run in polynomial time. G and K are similar to those in CTPS except that K is also responsible to generate* SA_1*'s secret/public key pair* (sk_{SA_1}, pk_{SA_1}).

(TD_{SA}, TP_{SA}) is a *Threshold Designation-Proxy* protocol between the original signer u_1 and the supervising agent SA_1. Both TD_{SA} and TP_{SA} take as input the public keys pk_1, pkg_2 and pk_{SA_1}, respectively. TD_{SA} also takes as input the secret key sk_1 of u_1, and TP_{SA} takes as input the secret key sk_{SA_1} of SA_1. As the result of the interaction, the expected local output of TP_{SA} is skp_{SA_1}, the *partial* proxy secret key of SA_1. The undeniability of both u_1 and SA_1 is achieved by adding sk_{SA_1} in the protocol.

$$[TD_{SA}(pk_1, pkg_2, pk_{SA_1}, sk_1),$$

$$TP_{SA}(pk_1, pkg_2, pk_{SA_1}, sk_{SA_1})] \rightarrow skp_{SA_1}.$$

(TD_u, TP_u) is a *Threshold Designation-Proxy* protocol between u_1 and $\{u_{2,j}\}_{n_2}$. u_1 runs TD_u to send warrant ω_1 to $\{u_{2,j}\}_{n_2}$. TP_u run by each proxy signer $u_{2,j}$ takes as input the public keys pk_1, pkg_2 and the secret key $sk_{2,j}$ respectively. As the result of the interaction, the expected local output of TP_u is $skp_{2,j}$, the *partial* proxy secret share of $u_{2,j}$. Note the difference of this $skp_{2,j}$ with that in *CTPS* scheme. We call it partial here because all $\{skp_{2,j}\}t_2$ can not perform anything without another part skp_{SA_1}.

$$[TD_u, TP_u(pk_1, pkg_2, sk_{2,j})] \rightarrow skp_{2,j}$$

(*CTD, CTP*) is a Chained Threshold Designation-Proxy protocol between U_2 and $\{u_{3,k}\}_{n_3}$. It is similar to that is defined in Definition 1.

$$[CTD(pk_1, pkg_2, pk_{SA_1}, pkg_3, \{skp_{2,j}\}t_2),$$

$$CTP(pk_1, pkg_2, pk_{SA_1}, pkg_3, sk_{3,k})] \rightarrow skp_{3,k}$$

CTPSwS is the (possibly) randomized *Chained Threshold Proxy Signing with Supervision* algorithm. It is run by U_3 with agreement of SA_1, and takes as input the t_3 out of n_3 corresponding partial proxy secret shares $\{skp_{3,k}\}t_3$ and SA_1's partial proxy secret skp_{SA_1} and a message $M \in \{0,1\}^*$, and outputs a chained threshold proxy signature with supervision $p\sigma_3$.

$$CTPSwS[\{skp_{3,k}\}t_3, skp_{SA_1}, M] \rightarrow p\sigma_3$$

CTPVwS is the deterministic *Chained Threshold Proxy Verification with Supervision* algorithm as follows.

$$CTPVwS[M, p\sigma_3, pk_1, pkg_2, pk_{SA_1}, pkg_3] = 0/1$$

CTPIDwS is the *Chained Threshold Proxy IDentification with Supervision* algorithm. It takes as input a valid proxy signature $p\sigma_3$, and outputs identities, i.e., public keys.

$$CTPIDwS(p\sigma_3) = [pkg_2, pk_{SA_1}, pkg_3] / \perp$$

Based on Definition 3, we give a draft of a concrete *CTPSwS* scheme here. Different from *CTPS*, u_1 needs to run TD_{SA} to calculate skt_1 and sends to his supervising agent SA_1. Different from Lui *et al.*'s idea, SA_1 runs TP_{SA} to generate $skp_{SA_1} = sk_{SA_1} e_1 + skt_1 \mod q$ using his secret key sk_{SA_1}. Without knowing $skt_{1,j}$, each $u_{2,j}$ runs TP_u to generate $skp_{2,j} = sk_{2,j}\, e_1 \mod q$. (*CTD*, *CTP*) run between

Copyright © 2009 SciRes

U_2 and $\{u_{3,k}\}_{n_3}$ is the same as that in *CTPS*. At last when each $u_{3,k} \in U_3$ agrees to the data part, they must forward this email to SA_1, Steven, for him to check the contract part. U_2 can not generate a valid proxy signature on behalf of Sam until Steven agrees to the contract part and contributes his partial proxy signature $ps_{SA_1} = skp_{SA_1} c_3 + k_{SA_1} \mod q$ where $k_{SA_1} \in_R Z_q$ and $r_{SA_1} = g^{k_{SA_1}} \bmod q$. Since SA_1's secret key sk_{SA_1} is involved, security level of our scheme is enhanced by adding SA_1-protected property.

5. Security of CTPSwS

The security model of *CTPSwS* is similar to that of *CTPS* defined in Definition 2, except that *A* can be the fourth role, *Role 4*: the supervising agent of u_1, SA_1. In terms of this role, the goal of *A* also includes a forgery *CTPSwS* by U_2, U_3 and SA_1 (*SHU*) on behalf of u_1.

Definition 4 (Security of CTPSwS) *Let* ***CTPSwS*** = {*G, K*, (*TD*$_{SA}$, *TP*$_{SA}$), (*TD*$_u$, *TP*$_u$), (*CTD, CTP*), *CTPSwS*, *CTPVwS, CTPIDwS*} *be a chained threshold proxy signature scheme. Consider an experiment* $\boldsymbol{Exp}_{CTPSwS,A}(k)$ *related to* ***CTPSwS****, adversary A and parameter k. In the extreme case, adversary A should represent all proxy signers and* SA_1 *if the SHU is the original signer, or the original signer,* SA_1*, and all other proxy signers except the SHU if the SHU is one of the proxy signers in level* 2 *or* 3*, or the original signers and all proxy signers if the supervising agent* SA_1 *is the SHU. Empty arrays* $\boldsymbol{skp}_{3,2}$*,* $\boldsymbol{skp}_{SA_1}$ *and empty sets* $\boldsymbol{pkg}_2$*,* $\boldsymbol{pkg}_3$ *are created. The adversary A can make the following requests and queries:*

R1: u_1(*SHU*) designates $\{u_{2,j}\}_{n_2}$ and $\{u_{3,k}\}_{n_3}$ as his proxy signer groups in level 2 & 3, respectively, and specifies SA_1 as u_1's supervising agent. *A* requests to interact with u_1(*SHU*) running *TD*$_{SA}$ and *TP*$_{SA}$, and plays roles of all others running(*TD*$_{SA}$, *TP*$_{SA}$) and (*CTD*, *CTP*). $\boldsymbol{pkg}_{SA_1}$ is set to $\boldsymbol{pkg}_{SA_1} \cup pkg_{SA_1}$.

R2: u_1 designates $\{u_{2,j}\}_{n_2}$ and $\{u_{3,k}\}_{n_3}$ as his proxy signer groups in level 2 & 3, where $u_{2,2}$ is the *SHU* and specifies SA_1 as u_1's supervising agent. *A* requests to interact with $u_{2,2}$ (*SHU*) running *TP*$_u$ and *CTD*, and plays roles of all others running (*TD*$_{SA}$, *TP*$_{SA}$), *TD*$_u$ and *CTP*. $\boldsymbol{pkg}_2$ is set to $\boldsymbol{pkg}_2 \cup pkg_2$.

R3: u_1 designates $\{u_{2,j}\}_{n_2}$ and $\{u_{3,k}\}_{n_3}$ as his proxy signer groups in level 2 & 3, respectively, and specifies SA_1 (*SHU*) as u_1's supervising agent. *A* requests to interact with SA_1 (*SHU*) running *TP*$_{SA}$, and plays roles of all others running the remanent algorithms and protocols. $\boldsymbol{skp}_{SA_1}$ is set to $\boldsymbol{skp}_{SA_1} \cup skp_{SA_1}$.

R4: u_1 designates $\{u_{2,j}\}_{n_2}$ and $\{u_{3,k}\}_{n_3}$ as his proxy signer groups in level 2 & 3, where $u_{3,2}$ is the *SHU*, and specifies SA_1 as u_1's supervising agent. *A* requests to interact with $u_{3,2}$ (*SHU*) running *CTP*, and plays roles of all others running the remanent algorithms and protocols. $\boldsymbol{skp}_{3,2}$ is set to $\boldsymbol{skp}_{3,2} \cup skp_{3,2}$.

Q1: Chained threshold proxy signature with supervision query, where $u_{3,2}$ is the *SHU*, on behalf of u_1. *A* can make a query (*M*,32,*x*) to oracle $O_{CTPVwS}(skp_{3,k}(k=1,3,\cdots,t_3), \boldsymbol{skp}_{3,2}[x], skp_{SA_1}, \cdot,\cdot,\cdot)$. If $\boldsymbol{skp}_{3,2}[x]$ has been defined, we say this query is valid and the oracle returns $p\sigma_3 = CTPSwS(skp_{3,k}(k=1,3,\cdots,t_3), \boldsymbol{skp}_{3,2}[x], skp_{SA_1}, M, \cdot,\cdot)$.

Q2: Chained threshold proxy signature with supervision query, where SA_1 is the *SHU*, on behalf of u_1. *A* can make a query (*M*,SA_1,*x*) to oracle $O_{CTPVwS}(skp_{3,k}(k=1,2,\cdots,t_3), \boldsymbol{skp}_{SA_1}[x], \cdot,\cdot,\cdot)$. If $\boldsymbol{skp}_{SA_1}[x]$ has been defined, we say this query is valid and the oracle returns $p\sigma_3 = CTPSwS(skp_{3,k}(k=1,2,\cdots,t_3), \boldsymbol{skp}_{SA_1}[x], M, \cdot,\cdot)$.

Eventually, *A* outputs a forgery $(M, p\sigma_3, pk_1)$. The output of the experiment is determined as follows:

- $CTPID_wS(p\sigma_3) \setminus \left(pkg_3 \cup pkg_{SA_1}\right) \notin \boldsymbol{pkg}_2$, or
- $CTPID_wS(p\sigma_3) \setminus \left(pkg_2 \cup pkg_{SA_1}\right) \notin \boldsymbol{pkg}_3$, or
- No valid query (*M*,32,*x*) to $O_{CTPVwS}(skp_{3,k}(k=1,3,\cdots,t_3), \boldsymbol{skp}_{3,2}[x], \cdot,\cdot,\cdot)$,
- No valid query (*M*,SA_1,*x*) to $O_{CTPVwS}(skp_{3,k}(k=1,2,\cdots,t_3), \boldsymbol{skp}_{SA_1}, \cdot,\cdot,\cdot)$.

Otherwise, the output is 0.

We define the advantage of adversary *A* as

$$\boldsymbol{Adv}_{CTPSwS,A}(k) = \Pr[\boldsymbol{Exp}_{CTPSwS,A}(k) = 1]$$

We say that ***CTPSwS*** is a secure chained threshold proxy signature with supervision scheme if the function $\boldsymbol{Adv}_{CTPSwS,A}(k)$ is negligible for *A* of time complexity polynomial in the security parameter *k*.

Security proof details follow the similar logic as the *CTPS* scheme, but are more tedious and skipped in this paper.

6. Conclusion and Discussion

This paper designs two provably secure schemes in random oracle model, Chained Threshold Proxy Signature (*CTPS*) scheme and Chained Threshold Proxy Signature with Supervision (*CTPSwS*) scheme, to solve the group-to-group delegation problem and delegation with supervision in delegation chain problem. They are very useful in email with signature system where a manager wants to delegate his signing right to his vice managers, who can further perform the delegation to their employees. In some cases, the delegatee's proxy signing rights can be supervised by manager's supervising agent. For future work, we hope to develop more flexible delegation

Copyright © 2009 SciRes

scheme that enables delegatees in different levels, say any one of vice managers and any two employees can cooperate to generate proxy signature. Also, more efficient schemes for these problems are always desirable.

REFERENCES

[1] Zoe L. Jiang, S. M. Yiu, L. C. K. Hui, Y. Dong, and S. H. Y. Wong, "Chained threshold proxy signature without and with Supervision," In 2008 International Conference on Computer Science and Software Engineering (CSSE'08), HongKong, China, pp. 837−840, December 2008.

[2] S. Kim, S. Park, and D. Won, "Proxy signatures, revisited," In 1st International Conference on Information and Communications Security (ICICS'97), LNCS 1334, Beijing, China, pp. 223−232, 1997.

[3] M. Mambo, K. Usuda, and E. Okamoto, "Proxy signatures for delegating signing operations," In 3rd ACM Conference on Computer and Communication Security, New Delhi, India, pp. 48−57, 1996.

[4] T. El Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," In IEEE Transactions on Information Theory, Vol. 31, No. 4, pp. 469−472, 1985.

[5] T. Okamoto, "Provably secure and practical identification schemes and corresponding signature schemes," In Advances in Cryptology (Crypto'92), LNCS 740, pp. 31−53, 1992.

[6] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," In Advances in Cryptology-Eurocrypt 1986 (EuroCrypt'86), LNCS 263, pp. 186−194, 1987.

[7] B. C. Neuman, "Proxy-based authorization and accounting for distributed systems," In Proceedings of 13th International Conference on Distributed Computing Systems, Pittsburgh, USA, pp. 283−291, May 1993.

[8] M. Cerecedo, T. Matsumoto, and H. Imal, "Efficient and secure multiparty generation of digital signatures based on discrete logarithms," In IEICE Transactions on Fundamentals of Electronics, Communications & Computer Science, Vol. E76-A, No. 4, pp. 532−545, 1993.

[9] R. W. C. Lui, L. C. K. Hui, and S. M. Yiu, "Delegation with supervision," In Information Sciences, Vol. 177, No. 19, pp. 4014−4030, 2007.

[10] A. Boldyreva, A. Palacio, and B. Warinschi, "Secure proxy signature schemes for delegation of signing rights," http://eprint.iacr.org/2003/096.

[11] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. "Proactive secret sharing or: How to cope with perpetual leakage," In Advances in Cryptology (Crypto'95), LNCS 963, Vol. 963, pp. 339−352, 1995.

[12] C. P. Schnorr, "Efficient signature generation by smartcards," In Journal of Cryptology, Vol. 4, No. 3, pp. 161−174, 1991.

Copyright © 2009 SciRes

APPENDIX

A Proof of Theorem 1

Suppose adversary A is a successful forger against CTPS scheme in polynomial time. Let adversaries B, C, and D wrap all communication channels from and to A, and have access to a standard signing oracle O_S, a chained threshold proxy signing oracle O_{CTPS}, and two random oracles functioning as hash functions G and H to answer A's requests and queries. First, system parameters params and secret/public key pairs are generated by running G and K. Empty array $\boldsymbol{wskp}_{3,2}$ and empty sets $\boldsymbol{pkg}_2$ and $\boldsymbol{pkg}_3$ are created. The adversary A is of all the secret keys except SHU's, and it can make the following requests and queries.

R1: A requests to interact with $u_1(SHU)$ running TD, and plays the role of $\{u_{2,j}\}_{n_2}$ running TP. The request is interrupted by B. B creates an appropriate warrant ω_1 and makes query $(0 \| pk_1 \| pkg_2 \| \omega_1)$ to its signing oracle $O_S(sk_1,\cdot)$. Upon receiving an answer (skt_1,r_1), it forwards (ω_1,skt_1,r_1) to $\{u_{2,j}\}_{n_2}$. After a successful run, $\boldsymbol{pkg}_2$ is set to $\boldsymbol{pkg}_2 \cup pkg_2$.

R2: A requests to interact with $\{u_{2,j}\}_{n_2}$ running TP, where $u_{2,2}$ is the SHU, and plays the role of u_1 running TD. C creates an appropriate warrant ω_2 and makes query $(0 \| pkg_2 \| pkg_3 \| \omega_2)$ to its signing oracle $O_S(skp_{2,2},\cdot)$. Upon receiving an answer $(skt_{2,2}, r_2)$, it divides the answer into n_3 shares using random polynomial $F_{2,2}(x)$ and forwards $(\omega_2, \{skt_{2,2,k}\}_{n_3}, r_2)$ to $\{u_{3,k}\}_{n_3}$. $\boldsymbol{pkg}_3$ is set to $\boldsymbol{pkg}_3 \cup pkg_3$.

R3: A requests to interact with $\{u_{3,k}\}_{n_3}$ running CTP, where $u_{3,2}$ is the SHU, and plays the role of U_2 running CTD. When A outputs ω_2, $skt_{2,j,2}$, r_2, D verifies them. If all the verifications pass, D stores $(\omega_2, skt'_{2,2}, r_2)$ in the last unoccupied position of $\boldsymbol{wskp}_{3,2}$.

Q1: A can make a query $(M,32,x)$ to oracle $O_{CTPS}(skp_{3,k}(k = 1,3,\cdots,t_3),\boldsymbol{skp}_{3,2}[x],\cdot,\cdot,\cdot)$. If $\boldsymbol{wskp}_{3,2}[x]$ is not defined, it returns $\perp$. Otherwise, D performs the following operations:

- Pick a random number $c_3 \in Z_q$.
- Pick a random number $ps_{3,2} \in Z_q$.
- Make a query $(0 \| pkg_2 \| pkg_3 \| \omega_2, r_2)$ to oracle G and let e_2 be the response.
- Compute commitment

$$r_{3,2} = g^{ps_{3,2}} (pk_{3,2}{}^{e_1} \cdot g^{skt_{2,j}})^{-c_3} \bmod p$$

- Set $H(1 \| M \| pkg_2 \| pkg_3 \| \omega_2 \| r_2, r_3) \leftarrow c_3$
- Return $(r_{3,2}, ps_{3,2})$ to A

Suppose after the experiment $\boldsymbol{Exp}_{CTPS,A}(k)$, A outputs a forgery in polynomial time of k such that at least one of the following events occurs:

E1. $CTPID(p\sigma_3) \setminus pkg_3 \notin \boldsymbol{pkg}_2$.

E2. $CTPID(p\sigma_3) \setminus pkg_2 \notin \boldsymbol{pkg}_3$.

E3. no valid query $(M,32,x)$ to $O_{CTPS}(skp_{3,k}(k = 1,3,\cdots,t_3),\boldsymbol{skp}_{3,2}[x],\cdot,\cdot,\cdot)$.

Assume **E1** occurs. Since all the coming in and out channels are wrapped by the adversaries, the query $H(1 \| M \| pkg_2 \| pkg_3 \| \omega_2 \| r_2, r_3)$ made by A must be grabbed and responded by c_3. By using forking lemma, B can rewind A to this point and gives A another random $c'_3 \neq c_3$. With non-negligible probability, A produces a forgery with respect to the same query, such that

$$g^{p\sigma_3} = PKP^{c_3} \cdot r_3 \bmod p,$$
$$g^{p\sigma'_3} = PKP^{c'_3} \cdot r_3 \bmod p.$$

Hence,

$$g^{(p\sigma_3 - p\sigma'_3) \bmod q} = PKP^{(c_3 - c'_3) \bmod q} \bmod p,$$
$$PKP = g^{SKP} \bmod p,$$
$$SKP = e_2 \cdot skg_3 + e_2(e_1(skg_2 + sk_1) + k_1) + k_2 \bmod q.$$

B subtracts $e_2 \cdot skg_3 + e_2(e_1 \cdot skg_2) + k_2$ for B knows skg_2 and skg_3. Then it obtains $\sigma'_1 = sk_1 \cdot e_1 + k_1 \bmod q$, a successful Schnorr signature of u_1.

Assume **E2** occurs. The deduction is similar to the above. However since the SHU is $u_{2,2}$, the adversary C should subtract the corresponding parts, and obtain $\sigma'_{2,2} = sk_{2,2} \cdot e_1 + k_1 \bmod q$, a successful Schnorr signature of $u_{2,2}$.

Assume **E3** occurs. The deduction is similar to the above. However since the SHU is $u_{3,2}$, the adversary D should subtract the corresponding parts, and obtain $\sigma'_{3,2} = sk_{3,2} \cdot e_2 + k_2 \bmod q$, a successful Schnorr signature of $u_{3,2}$.

Copyright © 2009 SciRes