INFORMATICA, 2004, Vol. 15, No. 3, 425–437

Proxy Confirmation Signatures

Chih-Hung WANG
Department of Computer Science and Information Engineering, National Chiayi University
Chiayi, Taiwan, R.O.C.
e-mail: wangch@mail.ncyu.edu.tw

Yen-Cheng CHEN
Department of Computer Science and Information Engineering, National Cheng Kung University
Tainan, Taiwan, R.O.C.
e-mail: cyc@ismail.csie.ncku.edu.tw

Received: January 2003

Abstract. The undeniable signature, introduced by Chaum *et al.* in 1989, provides a nice property that the signer has an additional control over who will benefit from being convinced by the signature. However, a conspicuous drawback of undeniable signature is that the signer may be unavailable or refuse to cooperate. Chaum in 1994 proposed a designated confirmer signature scheme to protect the recipient's right. There exists a confirmer, who can always help the recipient prove the validity of the signature to others. Unfortunately, Chaum's paper did not consider that a malicious confirmer proves the validity of the signature to any persons as his will or even leaks the sensitive information to the signer's enemies. This paper proposes a new signature scheme called proxy confirmation signature where the proxy confirmer can only acquire a *temporary proxy confirmation capability* instead of a perpetual one from the signer. That is, the signer not only can delegate the confirmation capability to the proxy confirmer, but also can revoke the proxy confirmer's capability for avoiding the abuse. Moreover, our scheme also provides a technique to properly restrict the proxy confirmer to convincing only some specified verifiers that the signature is valid.

Key words: cryptography, undeniable signatures, designated confirmer signatures, zero-knowledge proof, proxy confirmation signatures.

1. Introduction

Digital signature using public key cryptography is a widely applicable technology for electronic commerce. It provides message authenticity and a mechanism to solve the dispute between senders and receivers. An inborn feature of digital signature is that the signature can easily be copied and verified by everyone. Chaum and Antwerpen (1989) proposed that the proliferation of certified copies of the signature would be detrimental to the signer with blackmail attack or commercial espionage, and introduced a new concept called undeniable signature to solve this problem. The undeniable signature is different from the ordinary signature on requiring the signer's cooperation to verify the validity of the signature (Chaum, 1990; Chaum *et al.*, 1992; Gennaro *et al.*, 1997; Boyar *et al.*, 1991). The recipient can not convince others later by simply transmitting the copies of the

signature. Undeniable signature gives the signer additional control over who will benefit from being convinced by the signature; unfortunately, this property may be a limitation for many practical applications. If the signer should become unavailable or refuse to cooperate, then the recipient can not check the signature anyway.

Designated confirmer signatures (DCS), initially introduced by Chaum in 1994, eliminate the shortcoming of undeniable signature that the signature can only be verified by cooperating with the original signer. In Chaum's scheme, the signer S sends a signature to the recipient R and convinces R that this signature can be confirmed later by a designated confirmer C. Therefore, the recipient R would not worry about that the signer may refuse to cooperate because there is a confirmer C, can always help him prove the validity of the signature to an interested verifier V.

However, in DCS, the confirmer's confirmation capability is unlimited. Once the signer signs a confirmer signature, the confirmer has the confirmation capability perpetually and no one can terminate it or limit the confirmer's behavior (Chaum, 1994; Michels and Stadler, 1998; Nguyen *et al.*, 1999; Okamoto, 1994). For example, the signer may sign a terrible secret and fear that his enemies would find out he said this secret. But unfortunately, the conformer has the ability of proving the validity of the signature to the signer's adversaries. Accordingly, this paper mainly presents a new scheme called proxy confirmation signature (PCS), where a **proxy confirmer** can only acquire a *temporary proxy confirmation capability* from the signer. When the signer is unavailable to help the recipient prove the validity of the signature, he can delegate the confirmation capability on the signature to a proxy confirmer. Nevertheless, when the signer becomes available, he also can revoke the proxy confirmer's confirmation capability for avoiding the possible abuse.

The role of proxy confirmer (PC) in our scheme is quite different from the role of confirmer (C) in DCS scheme. C can be regarded as a trusted authority to take the place of the signer for the confirmation of the signature. But the PC in our scheme is only a *provisional agent* to help the signer confirm the signature. Therefore, our scheme cannot guarantee that the signature can be verified any time by the signer or confirmer; however, that is not a drawback. The feature of *confirmation revocation* in our scheme can be widely adopted in many e-commerce applications (see Section 5 for more description).

Outline. We organize the rest of this paper in the following. In Section 2, we give a basic model for proxy confirmation signature and some informal definitions used in our scheme. The construction of our scheme is presented in Section 3. In Section 4, we analyze the security and propose the improvements of the original scheme to prevent the conspiracy of the signer and proxy confirmer. Then we provide two applications to show the practicability of our new scheme in Section 5. Finally, the concluding remarks are given in Section 6.

2. Basic Model

DEFINITION 2.1 [*the proxy confirmation signature*]. Let S be a signer and PC be a proxy confirmer. A signature $Sign_{PCS}(S, PC, m)$ is called Proxy Confirmation Signature (PCS) on the message m if it satisfies the following requirements:

1. The signature $Sign_{PCS}(S, PC, m)$ can always be verified with the help of the original signer S.
2. If the signer S is unavailable to help the verifier prove the validity of the signature, he can delegate the confirmation capability to a proxy confirmer PC.
3. The signer can, at any time, revoke the proxy confirmer's confirmation capability by releasing some secrets. After that, the proxy confirmer no longer can help the verifier prove the validity of the signature.

Here, we briefly describe the concept of our signature scheme. Our scheme is divided into three phases. **(a) Signature Generating Phase:** The signer first creates a special undeniable signature where the signer must pre-determine a proxy confirmer before he signs the signature. **(b) Confirmation Delegating Phase:** If the signer cannot be available to perform the confirmation, he can delegate the confirmation capability on the signature to a proxy confirmer. After that, the proxy confirmer can prove the validity of the signature on behalf of the original signer. **(c) Revocation Phase:** The signer can revoke the proxy confirmer's confirmation capability on the signature by releasing a revocation key.

For reaching the goal of revocation, we need to apply a trap-door commitment function in which the disclosure of the secret would involve the destruction of the commitment function. This concept can be realized in our scheme that the signer constructs a proof for confirmation and keeps a secret for revocation. If the signer discloses the secret, the trap-door commitment function will be destroyed and no one can be convinced by the confirmation proof anymore. We briefly introduce the trap-door commitment and the proof of equality of the discrete logarithm in Definition 2.2 and Definition 2.3, respectively. The aspect of solution for revocable confirmation proof is shown in Definition 2.4.

DEFINITION 2.2 [*trap-door commitment* (also see (Brassard *et al.*, 1988) and (Jakobsson *et al.*, 1996))]. Let c be a function with input (y, u, v), where y denotes the public key of the user whose corresponding secret key is x, u is a value committed to and v is a random number. We say c is a trap-door commitment if and only if it satisfies the following requirements:

1. No polynomial algorithm, when given y, can find two different pairs of (u_1, v_1) and (u_2, v_2) such that $c(y, u_1, v_1) = c(y, u_2, v_2)$.
2. No polynomial algorithm, when given y and $c(y, u, v)$, can find u.
3. There exists an algorithm, when given x, (u_1, v_1) and a randomly selected number u_2, that can find v_2 such that $c(y, u_1, v_1) = c(y, u_2, v_2)$ (That means the user who knows the secret x, when given (u_1, v_1), can easily forge the committed value by substituting u_2 for u_1).

The following example was proposed by (Brassard *et al.*, 1988) and (Jakobsson *et al.*, 1996).

Let p and q be two large primes and $q|p-1$. The notation g denotes a generator of the subgroup, G_q, of Z_p^* of order q. The recipient's secret key is $x_R \in Z_q$ and the

corresponding public key is $y_R = g^{x_R} \bmod p$. The sender randomly selects $v \in Z_q$ and commits the value $u \in Z_q$ into c as:

$$c = g^u y_R^v \bmod p.$$

The sender sends (u, v) to the recipient for decommitting.

Jakobsson *et al.* (1996) proposed an efficient trap-door commitment scheme for multiple recipients P_i, $i = 1, 2, \ldots, n$. They modified the commitment to be $c = g^u (\prod_{i=1}^{n} y_i)^v \bmod p$, where y_i denotes P_i's public key. Each P_i would be convinced by the proof that u cannot be forged by others as long as he knows his secret key has not been compromised. Any other user would not gain this conviction because all P_i, $i = 1, \ldots, n$ can collude to cheat him.

DEFINITION 2.3 [*message-dependent proof of equality of the discrete logarithm* (Petersen, 1997)]. A message-dependent proof of equality of the discrete logarithm of y_1 to the base g_1 and y_2 to the base g_2 is a 2-tuple $(w, z) = Proof_{LogEQ}(m, g_1, y_1, g_2, y_2)$, where $w = F(m||g_1||y_1||g_2||y_2||{g_1}^z {y_1}^w||{g_2}^z {y_2}^w)$. This proof shows that the prover knows the discrete logarithm x: $\log_{g_1}(y_1) \equiv \log_{g_2}(y_2)$.

Prove Construction:

To construct this proof, the prover randomly selects $\kappa \in Z_q$ and calculates $w = F(m||g_1||y_1||g_2||y_2||{g_1}^\kappa||{g_2}^\kappa)$ and $z = \kappa - xw \bmod q$.

DEFINITION 2.4 [*message-dependent proof of equality of the discrete logarithm with trap-door commitment*]. Let X be a secret unknown to the prover and $Y = g^X \bmod p$ be a public value. A message-dependent proof of equality of the discrete logarithm of y_1 to the base g_1 and y_2 to the base g_2 with trap-door commitment is a 4-tuple $(w, z, u, v) = Proof_{LogEQTDC}(m, Y, g_1, y_1, g_2, y_2)$, where

$$w = F\big(m||T||g_1||y_1||g_2||y_2||{g_1}^z {y_1}^{(w+u)}||{g_2}^z {y_2}^{(w+u)}\big),$$

and $T = g^u Y^v \bmod p$ is a trap-door commitment. The prover, without knowing X, can use this proof to convince the verifier that he knows the discrete logarithm x: $\log_{g_1}(y_1) \equiv \log_{g_2}(y_2)$. On the other hand, the prover can easily create a forge transcript to cheat the verifiers if he knows X.

Prove Construction:

To construct this proof, the prover randomly selects $u, v, \kappa \in Z_q$ and calculates $T = g^u Y^v \bmod p$, $w = F(m||T||g_1||y_1||g_2||y_2||{g_1}^\kappa||{g_2}^\kappa)$ and $z = \kappa - x(w + u) \bmod q$.

3. Construction of Our Scheme

In the following, we describe our scheme with detailed steps.

System Setup. Let p be a large prime and q be a prime factor of $p - 1$ (i.e., $q|p - 1$). The notation g denotes a generator of subgroup, G_q, of ${Z_p}^*$ of order q, and F_1 and F_2

are two collision resistant hash functions. The secret key/public key pairs of the signer S, the proxy confirmer PC, and the verifier V are $(x_S, y_S = g^{x_S} \bmod p)$, $(x_{PC}, y_{PC} = g^{x_{PC}} \bmod p)$ and $(x_V, y_V = g^{x_V} \bmod p)$, respectively.

- **Signature Generating Phase:**
  - **Signing Protocol.** The signer first selects a random number $t \in Z_q$ and then computes $a = g^t \bmod p$ and $b = {y_{PC}}^t \bmod p$. The signer signs a special undeniable signature $\delta = (F_1(m||a)+b)^{x_S} \bmod p$ related to the proxy confirmer's public key.
  - **Confirmation by the Signer.** Since δ is an undeniable signature, the verifier can easily apply the general verification procedure (Chaum and Antwerpen, 1989) to verify the signature. The verifier randomly selects $e, d \in Z_q$ and then calculates $E = \delta^e (y_S)^d \bmod p$ and sends it to the signer. The signer calculates $L = E^{{x_S}^{-1}} \bmod p$ and sends it to V. Thus V can check if the equation $L = ((F_1(a||m)+b)^e g^d) \bmod p$ holds. However, according to the hinging method of (Chaum, 1994), the signer must prove the relation of a and b for avoiding the possible forging attacks. The signer S runs the interactive protocol of bi-proof $BP(g, a, y_{PC}, b)$ with V to show the discrete logarithm t: $\log_g(a) = \log_{y_{PC}}(b)$ (Fujioka *et al.*, 1992)(also see Appendix).
- **Confirmation Delegating Phase:**
  - **Delegation Protocol.** To delegate the proxy confirmer PC the capability of confirmation on the signature δ, the signer S randomly selects a secret X_δ and computes a public value $Y_\delta = g^{X_\delta} \bmod p$. S then randomly selects κ, u and v to construct a proof of $Proof_{LogEQTDC}(Y_\delta, g, y_S, F_1(m||a)+b, \delta) = (w, z, u, v)$, where $w = F_2(T||g||y_S||F_1(m||a)+b||\delta||g^\kappa||(F_1(m||a)+b)^\kappa)$, $T = g^u {Y_\delta}^v \bmod p$ and $z = \kappa - x_S(w+u) \bmod q$. Note that we eliminate the first parameter m in $Proof_{LogEQTDC}$ because the message has been included in other parameters: $F_1(m||a)+b$ and δ. After the signer releases the delegation key $K_\delta = (w, z, u, v)$, the proxy confirmer PC gains the confirmation capability on the signature δ.
  - **Confirmation by the proxy confirmer.** The proxy confirmer runs an interactive protocol of bi-proof $BP(g, y_{PC}, a, b)$ with V to show the discrete logarithm x_{PC}: $\log_g(y_{PC}) = \log_a(b)$. Besides, the verifier V needs to compute $T = g^u {Y_\delta}^v \bmod p$ and check $w \stackrel{?}{=} F_2(T||g||y_S||F_1(m||a) + b||\delta||g^z {y_S}^{w+u}||(F_1(m||a)+b)^z \delta^{w+u})$.
- **Revocation Phase:** When S wants to revoke PC's confirmation capability on the signature δ, he can release the secret X_δ. According to Definition 2.4, the proxy confirmer can construct a forge proof to cheat the verifier if he knows the secret X_δ. Thus, no one can be convinced by the proof of the proxy confirmer after the revocation phase. The detail analysis is shown in Theorem 4.1 at the next section.

REMARK 3.1. We address here that $(F_1(m||a) + b)$ needs to be a generator of G_q to meet the requirements of our scheme. Therefore, in signing procedure, the signer must

Signature Generating Phase: Signing Protocol

Signer: S Verifer: V

Selects a random number $t = 10 \in Z_q$.
Assume the message $m = 14$.
Computes:
$a = g^t \bmod p = 3^{10} \bmod 47 = 17$,
$b = {y_{PC}}^t \bmod p = 25^{10} \bmod 47 = 3$,
$\delta = (F_1(m||a) + b)^{x_S} \bmod p$
$= (F_1(14||17) + 3)^{17} \bmod 47 = (13 + 3)^{17} \bmod 47 = 24$.
Note that δ is a generator of G_q
because $\delta^q \bmod p = 24^{23} \bmod 47 = 1$.

$(a, b, \delta) = (17, 3, 24)$ →

Fig. 1. Example of Signature Generating Phase – Signing Protocol.

repeatedly select t and generate (a, b) until he finds a proper $(F_1(m||a) + b)$ such that $(F_1(m||a) + b)^q \equiv 1 \bmod p$. An alternative method is that we replace $(F_1(m||a) + b)$ with $(F_1(m||a) + b)^{\varepsilon}$ everywhere in our scheme if $p - 1 = \varepsilon q$. For example, in this case, δ needs to be modified to be $(F_1(m||a) + b)^{\varepsilon x_S} \bmod p$.

EXAMPLE 1. In the following, we deliver a simplified example with small numbers, which can help the readers have a better understanding of the proposed scheme. In practical application, the size of the numbers used in our scheme should be large enough to prevent the brute force attacks. The range of the prime p in our scheme is about from 512 bits to 1024 bits (i.e., $2^{512} \leqslant p \leqslant 2^{1024}$).

The system parameters are assigned as follows. Assume that $p = 47$ and $q = 23$ are primes and $q|p - 1$. The number $g = 3$ is a generator of G_q. The secret key/public key pairs of the signer S, the proxy confirmer PC and the verifier V are set to be $(17, 3^{17} \bmod 47 = 2)$, $(30, 3^{30} \bmod 47 = 25)$ and $(28, 3^{28} \bmod 47 = 8)$, respectively.

Since we only give an example of small numbers, it is inappropriate to compute the outputs of F_1 and F_2 by using a standard one-way hash function such as SHA or MD5. However, we can randomly assign a number to the output of hash function because it is reasonable and feasible to work in our example. Three procedures of our scheme are illustrated in Fig. 1 to Fig. 5.

4. Security Analysis and Scheme Improvements

Here, three security properties: unforgeability, indistinguishability and revocation, would be considered for our new scheme.

Unforgeability. This property means that the undeniable signature (a, b, δ) can not be forged because no one except S knows the secrets x_S. There are two possible attack

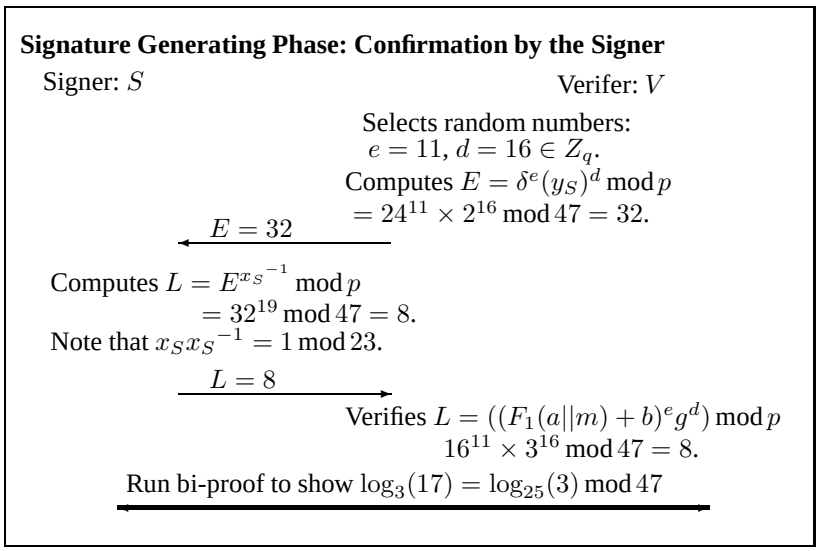

Fig. 2. Example of Signature Generating Phase – Confirmation by the Signer.

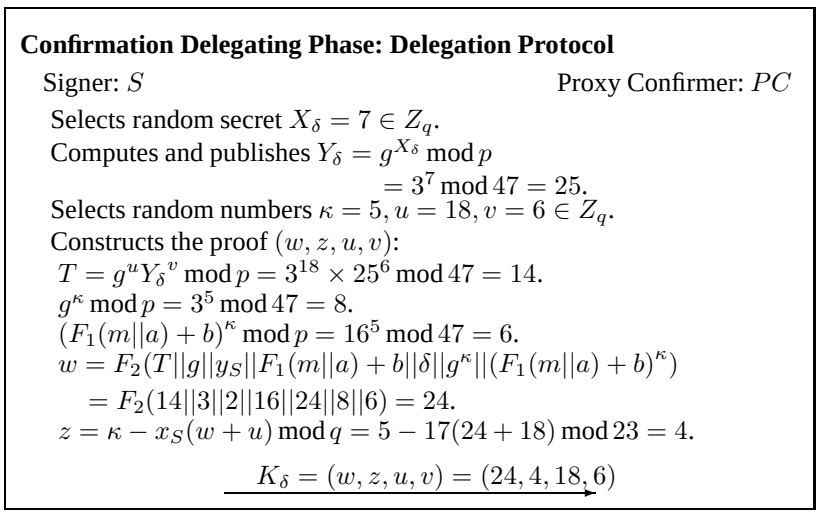

Fig. 3. Example of Confirmation Delegating Phase – Delegation Protocol.

scenarios that the intruder $\mathcal{I}$ tries to make a forgery signature (a^*, b^*, δ^*) without access to secret key x_S. The first one is that $\mathcal{I}$ selects a message m^*, a^* and computes $b^* = F_1(m||a) + b - F_1(m^*||a^*)$. However, the value of b^* which $\mathcal{I}$ can easily obtain would not have the same discrete logarithm as a^* has. That is because F_1 is a collision resistant hash function whose output is a random number. The second one is that $\mathcal{I}$ randomly selects t^* and also computes $a^* = g^{t^*} \bmod p$ and $b^* = {y_{PC}}^{t^*} \bmod p$. But $\mathcal{I}$ can not find a proper m^* satisfying $F_1(m^*||a^*) + b^* = F_1(m||a) + b$ because F_1 is computationally infeasible to be inverted.

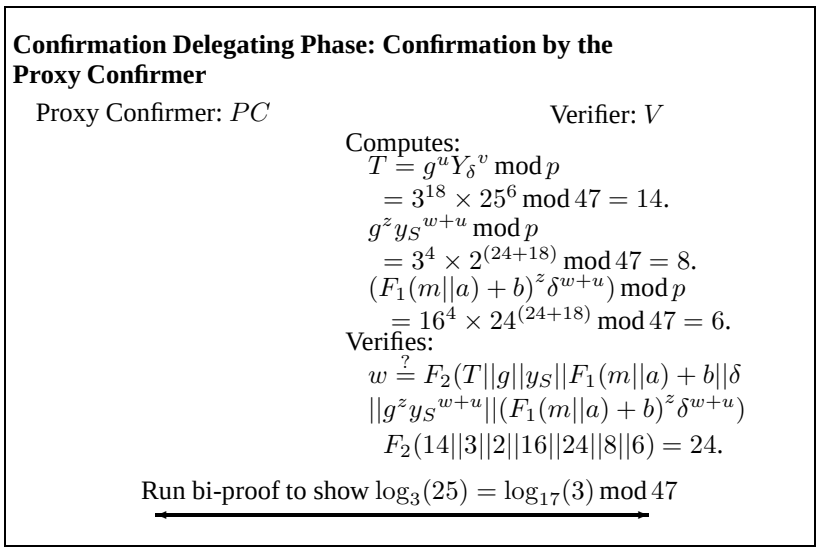

Fig. 4. Example of Confirmation Delegating Phase – Confirmation by the Proxy Confirmer.

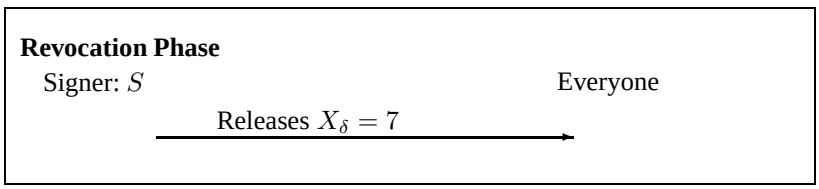

Fig. 5. Example of Revocation Phase.

Indistinguishability. The following lemma is used to analyze the property of indistinguishability in our scheme.

Lemma 4.1 [*Decision–Diffie–Hellman Assumption* (Michels and Stadler, 1998)]. *Let two sets be defined as follows:*

$$\chi = \big\{(g_1, g_2, y_1, y_2) \in G^4 | < g_1 >=< g_2 >= G\big\},$$

$$\mathcal{DH} = \big\{(g_1, g_2, y_1, y_2) \in \chi | \log_{g_1} y_1 = \log_{g_2} y_2\big\}.$$

Note that the elements of $\mathcal{DH}$ correspond to a Diffie–Hellman key exchange. For the base g_1, $g_2 = {g_1}^t$ and $y_1 = {g_1}^x$ are exchanged values. The resulting exchange key is $y_2 = {g_2}^x = {y_1}^t$. DDH assumption says that two random variables from χ and $\mathcal{DH}$, respectively, are computationally indistinguishable.

We assume that there exits an algorithm $\mathcal{A}$ that can distinguish a valid signature from a simulated one. We show that we can use $\mathcal{A}$ to solve the Decision–Diffie–Hellman problem. According to Lemma 4.1, assume that there are two signatures (plus the delegation

key) $\mathcal{S}_1 = (a, b, \delta, K_\delta)$ and $\mathcal{S}_2 = (a^*, b^*, \delta^*, {K_\delta}^*)$, where $\mathcal{S}_1 \in \mathcal{DH}$ is a valid signature and $\mathcal{S}_2 \in \chi$ is a simulated one. If we can use $\mathcal{A}$ to identify the correct signature from $\mathcal{S}_1$ and $\mathcal{S}_2$, then we can tell which pair, (a, b) or (a^*, b^*), has the same discrete logarithm, i.e., comes from $\mathcal{DH}$. This result violates the assumption of Lemma 4.1.

Given a^*, a simulated signature on the message m^* can be computed as $b^* = F_1(m||a) + b - F_1(m^*||a^*)$, $\delta^* = \delta$ and ${K_\delta}^* = K_\delta$. The verifier cannot distinguish the correct signature from the simulated signature because he knows nothing about the discrete logarithm of a^* to the base g and b^* to the base y_{PC}. Hence, without the confirmer's help, the verifier would not be convinced that both discrete logarithms of a^* and b^* are equal.

Revocation Property. The following theorem shows a main aspect of the revocation property in our scheme.

Theorem 4.1. *In the proxy confirmation signature scheme, if the signer releases the revocation key X_δ, the proxy confirmer no longer has the confirmation capability on the signature δ.*

Proof. If the signer releases the revocation key X_δ, everyone (including the proxy confirmer) can construct a simulated transcript by randomly selecting $\alpha, \beta, \theta, z \in Z_q$ and calculating:

$$
\begin{aligned}
&T = g^\alpha \bmod p,\\
&a = g^\theta \bmod p,\\
&b = {y_{PC}}^\theta \bmod p,\\
&w = F_2\Big(T||g||y_S||F_1\big(m^*||a\big) + b||\delta^*||g^z {y_S}^\beta||\big(F_1(m^*||a) + b\big)^z {\delta^*}^\beta\Big),\\
&u = (\beta - w) \bmod q,\\
&v = (\alpha - u)(X_\delta)^{-1} \bmod q.
\end{aligned}
$$

In the above transcript, (a, b, δ^*) is not a valid signature, but it can successfully pass the verification. Thus, no one can be convinced that the signature is valid by the proof of proxy confirmer on proving the relation of a and b.

TTP Involved. A security flaw indeed exists in the original scheme in Section 3. The signer knows the secret X_δ. If a malicious signer privately leaks X_δ to the proxy confirmer, the proxy confirmer can easily create a simulated transcript to cheat the verifier. Consequently, the verifier will not believe the confirmation by the proxy confirmer at all since he doubts that the conspiracy attacks of the signer and proxy confirmer may occur. Thus, we propose an improvement that needs a trusted third party (TTP) to create a secret/public value pair $(\bar{X_\delta}, \bar{Y_\delta} = g^{\bar{X_\delta}} \bmod p)$ for the signature δ. In the Confirmation Delegating Phase, the signer asks TTP announce a public key $\bar{Y_\delta}$, which can be used to construct the delegation key K_δ. In the Revocation Phase, the signer asks TTP release the corresponding secret key $\bar{X_\delta}$ to revoke the proxy confirmer's confirmation capability.

Restriction of the Proxy Confirmer's Capability. The signer can further restrict the proxy confirmer to convincing only some specified verifiers that the signature is valid. The following definition similar to Definition 2.4 can be used to extend our scheme.

DEFINITION 4.1 [*designated verifier message-dependent proof of equality of the discrete logarithm*]. Let V denote a designated verifier who has a secret key/public key pair $(x_V, y_V = g^{x_V} \bmod p)$. A designated verifier message-dependent proof of equality of the discrete logarithm of y_1 to the base g_1 and y_2 to the base g_2 is a 4-tuple $(w, z, u, v) = Proof_{DVLogEQ}(m, y_V, g_1, y_1, g_2, y_2)$, where $w = F(m||c||g_1||y_1||g_2||y_2||{g_1}^z {y_1}^{(w+u)}||{g_2}^z {y_2}^{(w+u)})$ and $c = g^u {y_V}^v \bmod p$ is a trap-door commitment. The prover, using this proof, only can convince the designated verifier V that he knows the discrete logarithm x: $\log_{g_1}(y_1) \equiv \log_{g_2}(y_2)$.

Prove Construction:

To construct this proof, the prover randomly selects $u, v, \kappa \in Z_q$ and calculates $c = g^u {y_V}^v \bmod p$, $w = F(m||c||g_1||y_1||g_2||y_2||{g_1}^\kappa||g_2^\kappa)$ and $z = \kappa - x(w+u) \bmod q$.

The above proof can be extended to multiple designated verifiers if the commitment c is changed to $c = g^u(\prod_{i=1}^n y_{V_i})^v \bmod p$, where y_{V_i} denotes the public key of the verifier V_i and n denotes the number of verifiers.

For achieving the purpose of restriction of the proxy confirmer's capability in our scheme, the signer in Confirmation Delegating Phase should construct $w = F_2(c||T||g||y_S||F_1(m||a) + b||\delta||g^\kappa||(F_1(m||a)+b)^\kappa)$, where $T = g^{u_1} {Y_\delta}^{v_1} \bmod p$, $c = g^{u_2}(\prod_{i=1}^n y_{V_i})^{v_2} \bmod p$ and $z = \kappa - x_S(w + u_1 + u_2) \bmod q$. The delegation key K_δ is changed to be $(w, z, u_1, u_2, v_1, v_2)$.

This improvement means only the verifiers belonging among the authorized set $\{V_i | i = 1, 2, \cdots, n\}$ can be convinced by the proof of the proxy confirmer. Therefore, the proxy confirmer has no ability to prove the validity of the signature to the person whom the signer does not allow to confirm this signature.

5. Applications

Here, we provide two applications to show that our improvement is valuable.

First, the proxy confirmation signature scheme can be used in software protection. The original concept was presented in the undeniable signature literature (Gennaro *et al.*, 1997). A software company can sign the undeniable signature on his published software and provide the confirmation service to the authorized users only. The users who illegally obtain the software from other users or Internet (i.e., the unauthorized users) cannot get the conviction that their copies are authentic. They will worry about that their copies may contain viruses or Trojan horse programs. Using the designated confirmer signature scheme in this application is much suitable for a large-scale company which has a great deal of customers, because the company can find several agents as the confirmers to help the customers to verify the software. That can greatly reduce the amount of the work of the company and improve the performance. However, the company has no way to handle

the problem that the agents dishonestly execute the confirmation procedure to prove the software authenticity to the unauthorized users. Therefore, for solving this problem, the software company can sign the PCS instead of DCS on the published software such that he can revoke the agent's confirmation capability when he finds that the agent violates the confirmation agreement.

Another application is that the PCS can be used in fair exchange protocol with off-line trusted third party (TTP). Chen has proposed an efficient fair exchange protocol by using DCS (Chen, 1998). We also can apply the PCS to fair exchange such that the TTP's confirmation capability can be limited or terminated to avoid possible abuses.

6. Conclusions

We have proposed a new method to realize the proxy confirmation on an undeniable signature. Our new scheme can properly prevent the misuse of the signature confirmation and provide both delegation and revocation protocols for the signer to restrict the capability of the proxy confirmer. In the ordinary signature, the proxy signing is an important and practical issue for many applications since the signer may be unavailable (Mambo *et al.*, 1996; Kim *et al.*, 1997). Similarly, in the undeniable signature, the proxy confirmation is essential to the protection of both signers and verifiers.

Appendix: Interactive Bi-proof of Equality

Fujioka *et al.* (1992) proposed an interactive bi-proof system that either proved $\log_\alpha(y) = \log_\beta(z)$ or proved $\log_\alpha(y) \neq \log_\beta(z)$. We use $BP(\alpha, y, \beta, z)$ to represent this proof system.

1. The verifier chooses random values u, $v \in Z_q$ and computes $a = \alpha^u y^v$, and sends a to the prover.
2. The prover chooses random values k, $\bar{k}$, $w \in Z_q$, computes $r_\alpha = \alpha^k$, $r_\beta = \beta^k$, $r_{\bar{\alpha}} = \alpha^{\bar{k}}$ and $r_{\bar{\beta}} = \beta^{\bar{k}}$, and sends r_α, r_β, $r_{\bar{\alpha}}$, $r_{\bar{\beta}}$ and w to the verifier.
3. The verifier sends u, v to the prover to open his commitment.
4. If $a \neq \alpha^u y^v$ then the prover halts, otherwise he computes $s = k - (v + w)x \bmod q$ and $\bar{s} = \bar{k} - (v + w)k \bmod q$, and also sends s, $\bar{s}$ to the verifier.
5. The verifier first checks whether $\alpha^s y^{v+w} = r_\alpha$, $\alpha^{\bar{s}} {r_\alpha}^{v+w} = r_{\bar{\alpha}}$ and $\beta^{\bar{s}} {r_\beta}^{v+w} = r_{\bar{\beta}}$, then he verifies:

   $$\beta^s z^{v+w} \stackrel{?}{=} r_\beta.$$

   If the above equation holds, then he concludes $\log_\beta(z) = \log_\alpha(y)$; otherwise $\log_\beta(z) \neq \log_\alpha(y)$.

Acknowledgement

This work was supported in part by National Science Council of Republic of China under contract NSC89-2218-E-214-020.

References

Boyar, J., D. Chaum, I. Damgard and T. Pedersen (1991). Convertible undeniable signatures. In *Advances in Cryptology – Proceedings of Crypto'90*, *Lecture Notes in Computer Science* (LNCS), **537**. Springer-Verlag. pp. 189–205.

Brassard, G., D. Chaum and C. Crepeau (1988). Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences*, **37**(2), 56–189.

Chaum, D., and H. van Antwerpen (1989). Undeniable signatures. In *Advances in Cryptology – Proceedings of Crypto'89*, *Lecture Notes in Computer Science* (LNCS), **435**. Springer-Verlag. pp. 212–217.

Chaum, D. (1990). Zero-knowledge undeniable signatures. In *Advances in Cryptology – Proceedings of Eurocrypt'90*, *Lecture Notes in Computer Science* (LNCS), **473**. Springer-Verlag. pp. 458–464.

Chaum, D. (1994). Designated confirmer signatures. In *Advances in Cryptology – Proceedings of Eurocrypt'94*, *Lecture Notes in Computer Science* (LNCS). Springer-Verlag. pp. 86–91.

Chaum, D., E. van Heijst and B. Pfitzmann (1992). Cryptographically strong undeniable signatures, unconditionally secure for the signer. In *Advances in Cryptology – Proceedings of Crypto'91*, *Lecture Notes in Computer Science* (LNCS), **576**. Springer-Verlag. pp. 470–484.

Chen, L. (1998). Efficient fair exchange with verifiable confirmation of signatures. In *Advances in Cryptology – Proceedings of Asiacrypt'98*. Springer-Verlag. pp. 286–299.

Fujioka, A., T. Okamoto, K. Ohta (1992). Interactive bi-Proof systems and undeniable signature schemes. In *Advances in Cryptology – Proceedings of Eurocrypt'91*, *Lecture Notes in Computer Science*. Springer-Verlag. pp. 243–256.

Gennaro, R., H. Krawczyk and T. Rabin (1997). RSA-Based undeniable signatures. In *Advances in Cryptology – Proceedings of Crypto'97*, *Lecture Notes in Computer Science* (LNCS), **1294**. Springer-Verlag. pp. 132–149.

Jakobsson, M., K. Sako and R. Impagliazzo (1996). Designated verifier proofs and their application. In *Advances in Cryptology – Proceedings of EuroCrypt'96*, *Lecture Notes in Computer Science* (LNCS), **1070**. Springer-Verlag. pp. 143–154.

Kim, S., S. Park and D. Won (1997). Proxy signatures revisited. In *Proc. of 1997 International Conference on Information and Communications Security* (ICICS'97). *Lecture Notes in Computer Science*, LNCS, **1334**. Springer-Verlag. pp. 223–232.

Mambo, M., K. Usuda and E. Okamoto (1996). Proxy signatures: delegation of the power to sign messages. *IEICE Trans. Fundamentals*, **E79-A**(9), 1338–1353.

Michels, M., and M. Stadler (1998). Generic constructions for secure and efficient confirmer signature schemes. In *Advances in Cryptology – Eurocrypt'98*, *Lecture Notes in Computer Science*. Springer-Verlag. pp. 406–421.

Nguyen, K., Y. Mu and V. Varadharajan (1999). Undeniable confirmer signature. *Information Security – Proceedings of Second International Workshop, ISW'99*, *Lecture Notes in Computer Science* (LNCS), **1729**. Springer-Verlag. pp. 235–246.

Okamoto, T. (1994). Designated confirmer signatures and public-key encryption are equivalent. In *Advances in Cryptology – Crypto'94*, *Lecture Notes in Computer Science* (LNCS), **839**. Springer-Verlag. pp. 61–74.

Petersen, H. (1997). How to convert any digital signature scheme into a group signature scheme. In *Proc. of Security Protocols Workshop'97*, LNCS, **1361**. Springer-Verlag. pp. 67–78.

C.-H. Wang was born in Kaohsiung Taiwan, in 1968. He received the BS degree in information science from Tunghi University and MS degree in information engineering from National Chung-Cheng University, Taiwan, R.O.C., in 1991 and 1993, respectively. He received the PhD degree in information engineering from National Cheng Kung University, Taiwan, R.O.C. in 1998. He is presently an assistant professor of Department of Computer Science and Information Engineering, National Chiayi University, Taiwan, R.O.C. His research interests include cryptography, information security and data compression.

Y.-C. Chen was born in Changhua, Taiwan, R.O.C., in 1975. He received his BS degree in information engineering from I-Shou University, Taiwan, in 1998 and his MS degree in information engineering from I-Shou University, Taiwan, in 2002. He is currently pursuing his PhD degree in the Department of Computer Science and Information Engineering, National Cheng Kung University, Taiwan. His research interests include cryptography and network security.

Įgaliojimo patvirtinimo parašai

Chih-Hung WANG, Yen-Cheng CHEN

Nepaneigiamas parašas, kurį 1989 m. pasiūlė Chaum *et al.*, pasižymi tokia gera savybe: pasirašęs asmuo turi papildomą galimybę kontroliuoti, kas gaus naudos iš įsitikinimo parašo tikrumu. Tačiau, nepaneigiamas parašas turi aiškiai matomą trūkumą: pasirašęs asmuo gali būti nepasiekiamas arba atsisakyti bendrauti. 1994 m. Chaum pasiūlė patobulintą (pažymėtą) patvirtintojo parašą, kad būtų galima apsaugoti gavėjo teises. Parašo patvirtintojas visada gali padėti gavėjui įrodyti parašo tikrumą kitiems. Tačiau, Chaum straipsnis nenagrinėja, kad piktavalis patvirtintojas norėdamas gali įrodyti parašo tikrumą bet kuriam asmeniui arba net perduoti svarbią informaciją pasirašiusiojo priešams. Šis straipsnis siūlo naują pasirašymo būdą, pavadintą įgaliojimo patvirtinimo parašu, kai įgaliojimo patvirtintojas, užuot įgijęs nuolatinį statusą iš pasirašiusiojo, įgauna tiktai laikiną parašo patvirtintojo statusą. Tai yra, pasirašęs asmuo ne tik gali deleguoti patvirtinimo galimybę įgaliojimo patvirtintojui, bet taip pat gali panaikinti įgaliojimo patvirtintojo galimybę, kad būtų galima išvengti piktnaudžiavimo. Be to, taikant pasiūlytą būdą galima tinkamai apriboti įgaliojimo patvirtinimą tik tam tikru iš anksto numatytų tikrintojų atžvilgiu.