A New Undeniable Signature with Partially Inheritable Confirmation Ability

（具有部分可繼承證實能力之不可否認簽章系統）

Chih-Hung Wang （王智弘）

Department of Information Management
I-Shou University Kaohsiung County,
Taiwan, R.O.C.
E-mail: wangch@isu.edu.tw

Yan-Cheng Chen（陳彥成）

Department of Information Management
I-Shou University Kaohsiung County,
Taiwan, R.O.C.
E-mail: m893323m@isu.edu.tw

中文摘要

1989 年 Chaum 等人提出了不可否認簽章的概念，簽章必須藉由與原簽章者合作才能進行驗證。之後出現了兩種不可否認簽章的替代架構：可轉換不可否認簽章[BCDP91]與指定證實者不可否認簽章[Cha94]，克服了原始不可否認簽章中，如果原簽章者臨時不在或者拒絕合作來協助驗證簽章的缺點。但是這兩種新的架構卻無法達到限制驗證者的目的，因為一旦這個簽章轉變成可自我驗證的簽章 (由簽章者或者證實者來轉換)之後，任何人都能夠確信簽章的正確。此外，在指定證實者不可否認簽章架構中，證實者在簽章產生的同時便能對任何人證明簽章的有效性。這說明簽章者尚無法有效地控制以下兩項：(1) 證實者何時擁有驗證簽章的能力與(2) 哪些驗證者可以獲得簽章的證明。因此必需在簽章者的控制能力上做更精確的設計，才可以更有效的應用在電子商務上。

本文將提出一個新的不可否認簽章系統，允許一個新的成員：「繼承者」來繼承簽章者的部分證實能力。新系統擁有的特性為：簽章者可以適當地決定哪些人可以從繼承者處獲得簽章的證明，以及繼承者何時可以擁有證實能力。我們新定義一個繼承金鑰 K_{inh} ，可以用來避免繼承者誤用證實的能力。亦即，在沒有得到簽章者給予的繼承鑰匙 K_{inh} 之前，證實者對簽章是沒有證實能力的。

Abstract

The concept of undeniable signature was introduced by Chaum et al. in 1989 [CA89], where the signature can only be verified by cooperation with the signer. Two alternative schemes, the convertible undeniable signature [BCDP91] and the confirmer signature [Cha94], can overcome the shortcomings of the original undeniable signature that a signer should become unavailable or refuse to cooperate. However, a critical problem of these two schemes is no limitation to the verifiers. Once the signature is converted to a self-authentication signature (by the signer or confirmer), anyone can be convinced that the signature is correct. Moreover, in the confirmer signature scheme, the confirmer can immediately prove the validity

of the signature to any verifier upon the signature being created; this means that the singer cannot flexibly control the *time* of delegating confirmation ability to the confirmer. Many applications require more precise design on the signer's control capability.

This paper will propose a new undeniable signature scheme that allows an additional party, called inheritor, to inherit the partial confirmation ability from the signer. Our scheme has the features that the signer can properly determine *who* can obtain the assurance from the inheritor that the signature is correct, and *when* the inheritor can acquire the confirmation ability. An i*nheritance key* K_{inh} defined in our scheme is used to prevent the confirmation ability from being misused by the inheritor. The inheritor has no ability to prove the signature until he receives K_{inh} from the signer.

Keywords： Undeniable Signatures, Designated Confirmer Signatures, Partially Inheritable Confirmation Ability, Zero-knowledge Proof.

1. Introduction

A conventional digital signature can easily be verified by everyone; therefore, the signer cannot deny its validity. However, this property of a conventional digital signature is not suitable for many practical applications on the electronic commerce. Consequently, the undeniable signature was introduced by Chaum and Van Antwerpen in 1989 [CA89], is a solution to this problem. Undeniable signature has three characteristic properties:

(1). The undeniable signature is only verified by cooperation with the signer, that preserves the non-repudiation property.

(2). The verifier cannot check the validity of a signature on his own.

(3). If the signer should become unavailable or refuse to cooperate, then the recipient cannot check the signature anyway.

Designated confirmer signatures, initially introduced by Chaum in 1994 [Cha94]. It eliminated the shortcomings of undeniable signatures that the signature is only verified by cooperating with the original signer. It is that verifiers can verify the validity of the signature by cooperation with the confirmer whom the signer designates. Presently, there are some confirmer signature studies being discussed generally in the literature [Cha94] [MS98] [NMV99] [CM00]. However, if we integrate the studies of the confirmer signature, some drawbacks would be presented as below:

(1). The signer will lose the highest authorization about the verification of the signatures. The confirmer can immediately prove the validity of the signature to any verifier upon the signature being created by the signer.

(2). It will weaken the protection of the verifying signature on the property of the undeniable signature. Under the situation, any verifiers will get the verification of the signature by cooperating with the confirmer. Another case is when the confirmer converts the signature to a self-authentication once; any verifiers can verify this signature without other persons' aid.

(3). Generally, the signer could lose the ability of the verifying signatures. In most

confirmer signature schemes, verifiers only verify the validity of the signature by cooperating with the confirmer.

It is an important issue on the designated confirmer signature to enhance the control capability of a signer. In our viewpoint, the signer must have the ability to determine *who* will benefit from being convinced by the signature, and *when* he would want to delegate his confirmation ability to others. This paper will propose a new scheme that allows an additional party, called inheritor, to inherit the partial confirmation ability from the signer. An inheritor can only inherit *partial* confirmation ability because he cannot prove the signature validity to the verifiers who is not per-determined by the signer. Further, we define an inheritance key K_{inh} created and kept by the signer, which is used to prevent the confirmation ability from being misused by the inheritor. As a result, the inheritor cannot prove the validity of the signature to any persons until he receives *inheritance key* K_{inh} from the signer.

The role of an inheritor in our new scheme is quite different from a confirmer in the designated confirmer signature scheme. Basically, in the designated confirmer signature scheme, the signer will no longer be responsible to prove the validity of the signature after he completes his signing procedure. Thus, the confirmer has the complete and independent confirmation ability without restriction. On the other hand, an inheritor in our scheme can only partially inherit the confirmation ability from the signer after he receives the inheritance key. The verifiers have been pre-determined and cannot be changed by the inheritor at all.

Our new scheme can be interpreted from a different aspect. That is, our scheme in fact provides a *conversion mechanism* for converting an undeniable signature to a confirmer signature with restrictive verifiers. If the inheritance secret has not been disclosed yet, our scheme would be an undeniable signature such that the verifiers should verify the signature by cooperation with the original signer. Once the inheritance secret is compromised, the signature will be completely transferred into a confirmer signature.

Give a last will example to explicate our new scheme. Suppose that the signer signs the last will and testament which contain some secret instructions that he would not want to publish. Further, the signer requires that a third party, unable to forge a legal will, can prove later that the signature is valid. A possible choice for the signer is to sign his will using a designated confirmer signature. After the signer's death, the confirmer can prove the validity of the signature to others. Unfortunately, A dishonest confirmer may prove the signature to signer’s enemies (ex. a competitor on the commerce) *before* or *after* the signer's death, and even he can arbitrarily convert the signature to a self-authentication one. Therefore, if our new scheme is used, the signer can previously commit the inheritance key to a trusted attorney. The inheritor will be given the inheritance key in the situation of signer’s death. After that, the inheritor can validate the signature in accordance with signer's volition.

We organize the rest of the paper as follows. We first describe the basic model and define

some used protocols in next section. We than construct our scheme in Section 3. After that, the security analysis of our new scheme will be presented in Section 4. The concluding remarks and suggestions for future work will be given in Section 5.

2. Preliminaries and Basic Model

For simplicity, we illustrate the notations that ***S*** represents the signer, ***I*** represents the inheritor and $\boldsymbol{V_i}$, $i \in [1,n]$ represents the verifiers pre-determined by the signer.

2.1 Basic Model

In the following, we give an informal definition to make our concept more precise.

Definition 1: (Undeniable Signature with Partially Inheritable Confirmation Ability).

A Signature $\alpha(S, I, \{V_i, i \in [1,n]\})$ is said to be an undeniable signature with partially inheritable confirmation ability if the inheritor I, after receiving the inheritance key from the signer S, can inherit the confirmation ability to validate the signature to the verifiers V_i, $i \in [1,n]$ pre-determined by S. We say that the inheritor only can inherit partial confirmation ability since he cannot change the verifiers being convinced by the signature.

According to the Definition 1, we briefly describe our scheme.

Signature Generating and Verification. The signer creates a signature $\alpha(S, I, \{V_i, i \in [1,n]\})$ and the confirmation ability can be delegated to an inheritor I. The signer generates the inheritance key K_{inh} and keeps it as a secret. For determining the verifiers who can be convinced by the signature, the signer employs the technique of *trap-door commitment* [BCC88, JSI96] (see Definition 2) to construct a *designated verifiers message-dependent proof of equality of the discrete logarithm* (see Definition 4). In addition, the signer must perform the verification protocol to guarantee that the signature is properly created.

Releasing the Inheritance Key. The inheritance key is an important factor for the inheritor to acquire the confirmation ability. The signer in this phase makes the inheritance key public such that each verifier can check the correctness of this key. Therefore, the verifiers can believe that there exists an inheritor who can help them verify the signature.

Proof by the Inheritor. The inheritor acquires the confirmation ability after he receives the inheritance key. He then runs an interactive proof to show the correctness of the signature.

2.2 Used Protocols

Definition 2: (Trap-Door Commitment) [BCC88, JSI96]

Let c be a function with input (y, u, v). The notation y denotes the public key of the user who has a corresponding secret x, and u is a value committed and v is a random number. We say c is a trap-door commitment if and only if it satisfies the following requirements:

1. Given y, no polynomial algorithm 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. Given y and $c(y, u, v)$, no polynomial algorithm can find u.

3. Given the secret x, (u_1, v_1) and a randomly selected number u_2, there is a polynomial algorithm that can find u_2 such that $c(y, u_1, v_1) = c(y, u_2, v_2)$ (This means the user who knows the secret x, given (u_1, v_1), can easily forge the committed value by changing u_1 into u_2).

The following example can be found in [BCC88, JSI96].

Example of Trap-door commitment. 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 secrets 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 the following : $c = g^u y_R{}^v \bmod p$

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

The scheme of multiple recipients can easily be constructed by computing n different c_i and r_i. However, there is an efficient scheme proposed by Jakobsson et al. in 1996 [JSI96].

Multiple Recipients Trap-Door Commitment. The commitment of u is modified to $c = g^u (\Pi_{i=1}^n y_R)^v \bmod p$.

Each V_i , $i = 1,2,3,\ldots,n$ would be convinced by the proof that u can not be forged by others as long as he knows his secret key has not been compromised. But any other user would not be convinced because all V_i , $i = 1,2,3,\ldots,n$ could collude to cheat him. The collusion would reveal the shared secret $\Sigma_{i=1}^n x_{R_i} \bmod q$ to all verifiers V_i and make the verifier V_i never believe the proof of commitment again.

Definition 3 : (Message-dependent Proof of Equality of the Discrete Logarithm) [Pet97]

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 tuple ：

$(w, z) = \text{Proof}_{\text{LogEQ}}(m, g_1, y_1, g_2, y_2)$,

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 : $\log_{g_1}(y_1) \equiv \log_{g_2}(y_2)$. To construct this proof, the prover randomly selects $k \in Z_q$ and calculates：

$$w = F(m \| g_1 \| y_1 \| g_2 \| y_2 \| g_1^k \| g_2^k) \text{ and}$$
$$z = k - x(w+u) \bmod q.$$

We combine Definition 2 and Definition 3 to construct the following proof that can be used to restrict the verifiers in our scheme.

Definition 4 : (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 four-tuple ：

$(w, z, u, v) = \text{Proof}_{\text{DVLogEQ}}(m, c, g_1, y_1, g_2, y_2, y_V)$,

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 : $\log_{g_1}(y_1) \equiv \log_{g_2}(y_2)$. To construct this proof, the prover randomly selects $u, v, k \in Z_q$ and calculates $c = g^u y_V{}^v \bmod p$,

$$w = F(m \| c \| g_1 \| y_1 \| g_2 \| y_2 \| g_1^k \| g_2^k) \text{ and}$$
$$z = k - x(w+u) \bmod q.$$

3、The Construction of Our Scheme

In our paper, instead of directly using the inheritor's public key y_I, we use $h = y_I^{\theta} \bmod p$, where $\theta = F_3(y_I \| m_{\text{inheritance}})^{x_S} \bmod p$, to construct our hinging method. The parameter θ is a part of inheritance key. This is, the inheritor needs to know the value of $x_I\theta$ if he wants to prove the equality of the discrete logarithm of $a = g^t \bmod p$ and $b = h^t = (g^{x_I\theta})^t \bmod p$.

Another important heuristic design of our scheme is to modify the random t into $t = F_3(y_S \| m)^{x_S} \bmod p$. In this case, the signer S can easily recover t when the signature is given for verification. Thus, S need not remember many different values of t to verify different signatures.

Our new scheme can be divided into the following procedures.

- **System Setup.** The parameters *p, q* and g are the same ones described in *Section 2*, and F_1, F_2 ,F_3 are tree collision resistant hash function. The secret key/public key pairs of the signer *S*, the inheritor *I*, the recipient *R* and the verifiers V_i , i = *1,2,3,…,n* are
  ($x_S, y_S = g^{x_S} \bmod p$),
  ($x_I, y_I = g^{x_I} \bmod p$),
  ($x_R, y_R = g^{x_R} \bmod p$) and
  ($x_{V_i}, y_{V_i} = g^{x_{V_i}} \bmod p$) $_{i=1,2,3,\ldots,n}$.

- **Signing Protocol.** Assume the signer has signed a undeniable signature α (a, b, δ) related to the inheritor's public key, i.e.,
  $\theta = F_3(y_I \| m_{\text{inheritance}})^{x_S} \bmod p$,
  $t = F_3(y_S \| m)^{x_S} \bmod p$,
  $h = y_I^{\theta} \bmod p$,
  $a = g^t \bmod p$,
  $b = h^t \bmod p$ and
  $\delta = (F_1(m \| a \| m_{\text{inheritance}}) + b)^{x_S} \bmod p$,
  where $m_{\text{inheritance}}$ denotes a warrant which contains the identity of the inheritor and the information about the inheritor's right. For delegating *I* the ability of confirming this signature, the signer randomly selects *k*, *u*, *v* and constructs a proof of
  $(w, z, u, v) = \text{Proof}_{\text{DVLogEQ}}(c, g, y_S,$
  $F_1(m \| a \| m_{\text{inheritance}}) + b, \delta, y_{V_i \{i=1,2,\ldots n\}})$
  where $c = g^u (\Pi_{i=1}^n y_{V_i})^v \bmod p,$
  $w = F_2(c \| g \| y_S \| F_1(m \| a \| m_{inheritance}) + b \| \delta \| g^k$
  $\| (F_1(m \| a \| m_{inheritence}) + b)^k)$
  $z = k - x_S(w + u) \bmod q$
  Thus, the signature signed by *S* denotes $\alpha(S, C, V_i, i=1,2,\ldots n) = (a, b, u, v, w, z, \delta, m_{\text{inheritance}})$.
  The *inheritance key* K_{inh} = $(m_{\text{inheritance}}, \theta)$ is kept as a secret by *S* .

- **Proof by the Signer.** Vi can easily check the proof by computing $c = g^u (\Pi_{i=1}^n y_{V_i})^v \bmod p$ and verifying
  $w \overset{?}{=} F_2(c \| g \| y_S \| F_1(m \| a \| m_{inheritance}) + b \| \delta$
  $\| g^z y_S^{(w+u)} \| F_1(m \| a \| m_{inheritance}) + b)^z \delta^{(w+u)})$.
  For proving the relation of *a* and *b*, *S* first computes $t = F_3(y_S \| m)^{x_S} \bmod p$ and runs the interactive protocol of bi-proof *BP*(*g,a,h,b*) [FOO92] with the V_i to show the discrete logarithm t: $\log_g(a) = \log_h(b)$ (The reader can refer to the appendix for detail).

- **Releasing the Inheritance Key.**
  After the signer releases the inheritance key, V_i can check $h \overset{?}{=} y_I^{\theta} \bmod p$.
  If the above equation holds and the signature proof by the signer is valid, V_i can be convinced that there exists an inheritor *I* who can prove the validity of

the signature.

- **Proof by the inheritor.** After getting the inheritance key $(m_{\text{inheritance}}, \theta)$, inheritor will inherit the confirmation ability to verify the signature. Therefore, V_i checks whether the signature $(a,b,u,v,w,z,\delta,m_{\text{inheritance}})$ is created properly. The inheritor runs the interactive protocol of bi-proof $BP(g,h,a,b)$ to show the discrete logarithm $x_1\theta: \log_g(h) = \log_a(b)$. Since $b = h^t = (g^{x_1\theta})^t \bmod p$, the inheritor cannot complete this proof without knowing the value of θ .

- **Conversion Protocol.** The inheritor I can convert the signature into a self-authentication signature that can only convince the verifiers $V_i, i \in [1,n]$ pre-determined by S. Here, I randomly select $\sigma \in Z_q$ and computes $T = \sigma + x_I F(a,r) \bmod q$, where F is a hash function. The inheritor I sends (σ, T) to all V_i , thus, V_i can verify $a^T \stackrel{?}{=} rb^{F(a,r)}$ [Cha94].

4. Security Analysis and Discussion

Here, three security properties would be considered for our new scheme.

(1). Unforgeability of signature: There exists no polynomial time algorithm which can forge the undeniable signature $\alpha(a,b,\delta)$ unless one except the signer S knows the secret key x_S . There are two scenarios that an attacker ***A*** tries to forge $\alpha^*(a^*, b^*, \delta^*)$ without access of the secret key x_S . The first one is that ***A*** selects a message m^*, a^* and computes $b^* = F_1(m \| a \| m_{\text{inheritance}}) + b - F_1(m^* \| a^* \| m_{\text{inheritance}})$. However, though ***A*** can easily obtain b^*, it would not have the same discrete logarithm as a^* has because F_1 is a collision resistance hash function which outputs a random number. The second one is ***A*** randomly selects t^* and compute $a^* = g^{t^*}$ and $b^* = y_C^{t^*}$. But ***A*** cannot find a proper m^* satisfying $F_1(m^* \| a^* \| m_{\text{inheritance}}) + b^* = F_1(m \| a \| m_{\text{inheritance}}) + b$ because F_1 is infeasible to be inverted.

(2). Indistinguishability of signature: Given a^*, a simulated signature on the message m^* is computed as $b^* = F_1(m \| a \| m_{\text{inheritance}}) + b - F_1(m^* \| a^* \| m_{\text{inheritance}})$ and $\delta^* = \delta$. The verifier cannot distinguish between the correct signature and simulate signature because he doesn't know anything about the discrete logarithm of a^* to the base g and b^* to the base y_I . Hence, without the signer's or inheritor's help, the verifier would not be convinced that both discrete logarithm of a^* and b^* are equal.

(3). Consistency of verification: For the signature $(a,b,u,v,w,z,\delta,m_{\text{inheritance}})$, the inheritor can help R verifying the signature to verifiers V_i , $i = 1,2,3,\ldots,n$ by running the interactive protocol of bi-proof $BP(g,h,a,b)$ between I and V_i to show the discrete logarithm $t: \log_g(a) = \log_h(b)$. The inheritor cannot claim that a correct (incorrect) signature is incorrect (correct).

(4). Simulating by the Designated Verifiers: We specially discuss the designated verifier's

process of simulation. Colluding of all V_i without accessing the signer's secret x_S can easily do the following simulating transcripts of the proofs. This means the verifiers not belong among V_i, $i = 1,2,3,\ldots,n$ would not be convinced by the proof even if the inheritor proves that a and b has the same discrete logarithm corresponding to the base g and y_I. The designated verifiers can collude to simulate the correct transcripts by randomly selecting $\alpha, \beta, \tau, z \in Z_q$ and calculate:

$$c = g^{\alpha} \bmod p$$
$$a = g^{\tau} \bmod p$$
$$h = {y_I}^{\alpha} \bmod p$$
$$b = h^{\tau} \bmod p$$
$$w = F_2(c \| g \| y_S \| F_1(m^* \| a \| m_{\text{inheritance}}) + b \| \delta^* \| g^Z {y_S}^{\beta} \| F_1(m^* \| a \| m_{\text{inheritance}}) + b))^Z \delta^{*\beta})$$
$$u = (\beta - \omega) \bmod q$$
$$v = (\alpha - u)(\sum_{i=1}^{n} x_{V_i})^{-1} \bmod q$$

5. Conclusions

We have proposed an undeniable signature scheme with partially inheritable confirmation ability which provide a nice approach to convert an undeniable signature to a confirmer signature. In our scheme, the signer can easily determine the time of delegating the confirmation ability to the inheritor. However, if the signer refuse to release the inheritance key, the inheritor is impossible to perform the confirmation procedure. To overcome this drawback is still an open problem, which we will investigate in the future.

Acknowledgement

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

6. References

[BCC88] G. Brassard, D. Chaum, C. Crepeau. Minimum Disclosure Proofs of Knowledge. Journal of Computer and System Sciences, Vol. 37, No. 2, pages 156-189, 1988.

[BCDP91] J. Boyar, D. Chaum, I. Damgard and T. Pedersen. Convertible Undeniable Signatures. In Advances in Cryptology - proceedings of Crypto'90, Lecture Notes in Computer Science (LNCS) 537, pages 189-205, Springer-Verlag, 1991.

[CA89] David Chaum and Hans Van Antwerpen: Undeniable Signature. In Advances in Cryptology - proceedings of Crypto'89, Lecture Notes in Computer Science (LNCS) 435, pages 212-217, Springer-Verlag, 1989.

[Cha94] D. Chaum. Designated Confirmer Signatures. In Eurocrypt'94, pages 86-91, 1994.[Jak94] M. Jakobsson. Blackmailing Using Undeniable Signature. In Eurocrypt'94, pages 425-427, 1994.

[CM00] Jan Camenisch, Markus Michels. Confirmer Signature Schemes Secure against Adaptive Adversaries. In Advances in Cryptology - proceedings of Eurocrypt'2000, Lecture Notes in

Computer Science(LNCS)1807, pages 243-258, Springer-Verlag, 2000.

[FOO92] A. Fujioka, T. Okamoto, K. Ohta. Interactive Bi-Proof Systems and Undeniable Signature Schemes. In Advances in Cryptology - proceedings of Eurocrypt'91, Lecture Notes in Computer Science, pages 243-256, Springer-Verlag, 1992.

[JSI96] M. Jakobsson, K. Sako and R. Impagliazzo. Designated Verifier Proofs and Their Application. In Advances in Cryptology - proceedings of EuroCrypt'96, Lecture Notes in Computer Science (LNCS) 1070, pages 143-154, Springer-Verlag, 1996.

[MS98] M. Michels and M. Stadler. Generic Constructions for Secure and Efficient Confirmer Signature Schemes.In Advances in Cryptology - Eurocrypt'98, Lecture Notes in Computer Science, pages 406-421, Springer-Verlag, 1998.

[NMV99] K. Nguyen, Y. Mu, and V. Varadharajan. Undeniable Confirmer Signature. Information Security - Proceedings of Second International Workshop, ISW'99, Lecture Notes in Computer Science (LNCS) 1729, pages 235-246, Springer-Verlag, 1999.

[Pet97] H. Petersen. How to Convert any Digital Signature Scheme into a Group Signature Scheme, to appear in Security Protocol'97, LNCS, Spring Verlag, 1997.

Appendix : Interactive Bi-proof of Equality

Fujioka et. Al. in 1992 proposed an interactive *bi-poof* system [FOO92] that either proved $\log_g(a) = \log_y(b)$ or $\log_g(a) \neq \log_y(b)$. This proof system can be used to construct our scheme on the inheritor proving the correctness of the signature to the pre-determined verifiers. We use $BP(g,a,y,b)$ to represent this proof system.

1. The verifier chooses random values $u, v \in Z_q$ and computes $d = g^u a^v$, and sends d to the prover.
2. The prover chooses random values $k, \bar{k}, w \in Z_q$, computes $r_g = g^k, r_y = y^k, \bar{r}_g = g^{\bar{k}}$ and $\bar{r}_y = y^{\bar{k}}$, and sends $r_g, r_y, \bar{r}_g, \bar{r}_y$ and w to the verifier.
3. The verifier sends u,v to the prover to open his commitment.
4. If $d \neq g^u a^v$ then the prover halts, otherwise he computes $S = k - (v+w)x \mod q$ and $\bar{s} = \bar{k} - (v+w)k \bmod q$, and sends $S, \bar{S}$ to the verifier.
5. The verifier first checks whether $g^s a^{v+w} = r_g, g^{\bar{s}} r_g^{v+w} = \bar{r}_g$ and $y^{\bar{s}} r_y^{v+w} = \bar{r}_y$, then he verifies: $y^s b^{v+w} \stackrel{?}{=} r_y$.

   If the above equation holds, then $\log_g(a) = \log_y(b)$, otherwise $\log_g(a) \neq \log_y(b)$.