JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 22, 1389-1407 (2006)

Practical Constructions to Multiple Designated Confirmer Signatures*

CHIH-HUNG WANG
Department of Computer Science and Information Engineering
National Chiayi University
Chiayi, 600 Taiwan

Designated confirmer signatures, initially introduced by Chaum in 1994, eliminated the undeniable signature shortcoming in which the signature can only be verified through the cooperation of the original signer. This paper proposes several practical constructions to multiple designated confirmer signature schemes. We employ the *message- dependent proof of equality of the discrete logarithm* and *trap-door commitments* to construct a new type of multiple confirmer scheme, where specified verifiers designated by the signer would have an advantage in verifying a signature where only one (out of *n*) confirmer is needed for a confirmation. Other verifiers, who the signer did not designate, must ask all *n* confirmers to collaboratively confirm this signature. This paper also provides an efficient and practical solution in applying the threshold functions on the designated confirmer signatures.

Keywords: cryptography, undeniable signature, designated confirmer signatures, zero-knowledge proof, threshold signature

1. INTRODUCTION

The undeniable signature concept was introduced by Chaum *et al*. in 1989 [7]. It is different from the ordinary signature in that it requires the signer's cooperation to verify the validity of the signature [8, 11, 21]. No one would be able to convince other people at a later time by simply copying the signature or the contents of interactive proofs between the signer and the recipient. The undeniable signature provides suitable technology in giving the signer additional control over who will be able to benefit from being convinced by a signature. Unfortunately, this property may be a limitation for many practical applications. If the signer becomes unavailable or refuses to cooperate, then the recipient will not be able to check the signature.

Designated confirmer signatures, initially introduced by Chaum in 1994 [10], eliminated the shortcoming of the undeniable signature in which the signature can only be verified through the cooperation of the original signer. In the signing phase, the signer *S* sends a signature on the message *m* to the recipient *R* and convinces *R* that this signature can be confirmed by a designated third party *C*. The recipient *R* cannot forge the signer's signature or convince others in a later time that the signature was truly signed by *S*, even if *R* records the interactions of the signing procedure. However, the designated third party

Received September 8, 2004; revised March 3, 2005; accepted May 5, 2005.
Communicated by Randy Chow.
* This work was supported in part by the National Science Council of Taiwan, R.O.C., under contract No. NSC 89-2218-E-214-020.

C, in the confirmation protocol, can always help *R* prove the validity of the signature to any verifier *V*.

In increasing availability and security, constructing the extension schemes to multiple designated confirmer signatures is an important issue. Basically, the confirmation protocol implemented with only one confirmer may create both performance and security bottlenecks [24]. For example, a malicious confirmer could violate the promise with the signer to show the validity of the signature to the signer's enemies (the signer may sign a terrible secret and fear that his enemies will find out he exposed this secret). A confirmer could fail, have a mistake, or even refuse to help some specified recipients to verify a signature. Generally, there are three models of the multiple confirmer scheme: (a) all *n* confirmers' collaboration is needed for a confirmation, (b) any one (out of *n*) confirmer can help the recipient verify the signature, and (c) any *k* out of *n* confirmers can help the recipient verify the signature. The model-(a) distributes the capability of confirmation to *n* confirmers, so this model can only enhance security. The model-(b) replicates *n* confirmers such that each one has the complete capability to confirm the signature, so this model can only enhance reliability. However, model-(c) is a highly reliable and highly secure scheme which has the following properties: less than a threshold number of colluding confirmers cannot compromise security (e.g. illegally help the recipient verify the signature); the signature can be verified if a threshold number of uncompromised confirmers are available and agree to run confirmation protocols.

Contributions. In this paper, we propose several practical solutions to the multiple confirmer schemes. We especially focus on combining model-(a) and model-(b) to create a new type of multiple confirmer scheme named *dual*{(1, *n*), (*n*, *n*)}, where some specified verifiers designated by the signer would have an advantage in verifying a signature in which only one (out of *n*) confirmer is needed for a confirmation. Other verifiers, who the signer did not designate, must ask all *n* confirmers to collaboratively confirm a signature. Assuming that the signer pre-determined a group of verifiers $\Lambda = \{V_i \mid i = 1, 2, \ldots, s_d\}$, this new *dual*{(1, *n*), (*n*, *n*)} scheme could be highly reliable to the specified verifiers $V_i \in \Lambda$. Any one (out of *n*) confirmer can convince the verifiers $V_i \in \Lambda$ that the signature is valid, whereas he cannot convince the verifiers $V_i' \notin \Lambda$. The techniques of *message-dependent proof of equality of the discrete logarithm* [33] and *trap-door commitments* [2, 27] are used to achieve our goal.

We also present a model-(c) scheme that uses the technique of *Verifiable Secret Sharing* (*VSS*) [16, 32] to add the threshold functions into the multiple confirmer signature scheme. A *Verifiable Discrete Logarithm Shares Construction Proof* (*VDLSCP*), which is very similar to the VSS protocol but has some different purposes, would be developed in this scheme. In *VDLSCP*, everyone can check that the discrete logarithm of a public value a_i is a proper share of a random secret d_0. Our solution is efficient and would greatly reduce the communication cost and size of the signature.

Applications. Our proposed multiple confirmer signature scheme offers a valuable contribution and is worth further research, since it provides several solutions to fulfill the requirements of many applications in practice. The proposed scheme has some applications that would be useful in electronic commerce. The first example is a fair exchange

protocol (usually used for network purchasing scheme or contract signing protocol). The designated confirmer signature is a tool for designing a fair exchange protocol of signatures with an off-line trusted third party (off-line TTP). Chen [13] in 1998 proposed an efficient fair exchange protocol by using confirmation signature scheme. Boyd and Foo [5] further proposed that the fair exchange protocol can be constructed by any convertible signature. In 1999, Garay *et al*. proposed an abuse-free optimistic contract signing protocol [20]. They employed a type of signature called a *private contract signature* (PCS) in designing their protocol (our analysis shows that the PCS can be implemented by a variant of the designated confirmer signature or by other convertible signatures). Our solution to multiple confirmer signatures can be used to improve security as well as the reliability of the fair exchange protocol. We propose a (*k*, *n*) threshold scheme that provides flexible integration strategies, since the security and reliability of a system are usually a trade-off. The detailed description is given in section 5.

The second example is that the proposed scheme can be applied to software protection. The original concept was proposed by Gennaro *et al*. in the RSA-based undeniable signature scheme [21]. A software company can sign its products using our scheme to convince authorized users that the released programs are valid and do not contain viruses or Trojan horses. Therefore, an unauthorized user who has an illegal copy cannot get this confirmation. Besides, a large-scale company which has many branches can delegate the confirmation capability to multiple confirmers in order to provide a convenient and rapid service to its users.

Related Work. Several designated confirmer signatures have been previously proposed. Okamoto in 1994 [31] proposed that a secure designated confirmer signature scheme can exist if and only if a secure public-key encryption scheme exists. In his paper, some practical constructions based on Schnorr and extended Fait-Shamir schemes were also presented. However, his paper did not reveal how a multiple designated confirmer signature scheme is designed.

Michels and Stadler in 1998 [29] proposed generic constructions for confirmer signature schemes. They introduced a new tool called confirmer commitment, which delegated the confirmation ability to a designated confirmer. The multiple confirmer construction concept was discussed in their paper; however, no specified method on dealing with threshold functions was presented. The detailed comparison between our proposed scheme and Michels and Stadler's scheme will be shown in a later section.

Some of the recent studies of confirmer signature focused on the *invisibility* property. Camenisch and Michels in 2000 pointed out that many previous models were vulnerable to an adaptive signature-transformation attack [6]. Assuming that the confirmer is disallowed to help a certain malicious user to confirm the signature δ, the malicious user can use this kind of attack to transfer the original confirmer signature to an independent form $\hat{\delta}$ by choosing his own signature public and secret keys, such that $\hat{\delta}$ is valid only if δ is valid. Owing to the indistinguishability between a transfered signature and a really new signature, the confirmer may reply to the malicious user as to whether the signature $\hat{\delta}$ is valid or not. As a consequence, the malicious user in fact knows whether δ is valid. Galbraith and Mao in 2003 further proposed an RSA-based confirmer signature that is

secure against adaptive adversaries [19]. We have not considered this adaptive attack in the proposed schemes and plan to develop new models in future research. However, some practical applications of the confirmer signature may reduce the possibility of performing this attack. For example, in the fair exchange protocol, the signer can obtain the conversion service by TTP only if he presents the complete evidence of the transaction and provides his own signature for exchange.

There are also some transformation models that have been proposed. Goldwasser and Waisbard proposed an efficient transformation of a large class of digital signature schemes into secure designated confirmer signature schemes [23]. They prove that the resulting designated confirmer signature transformed by their model can be provably secure under the same assumptions made by the originally used digital signature scheme and the encryption scheme. Another model suggested in [37] use DSA and RSA in the designated confirmer signature.

Some researchers attempted to develop a confirmer signature from bilinear pairings. Han *et al.* [25] claimed that they had an identity-based confirmer signature scheme. However, their scheme is simply a deniable signature and can be broken by a denial attack and a forge attack [38]. We plan to design a pairing-based confirmer signature which can be easily extended to multiple confirmer schemes.

Outline. Section 2 gives several definitions and techniques used in our schemes. In section 3, we show how to delegate the verifying signature ability to multiple confirmers, and propose several practical solutions to the multiple confirmer signature schemes. In section 4, we analyze security and discuss the properties of our proposed schemes. The application of the fair exchange protocol is shown in section 5. We make a comparison between our scheme and previous work in section 6. Concluding remarks and future researches are given in section 7.

2. PRELIMINARIES

2.1 Basic Model

We describe our basic model by using the following definitions.

Definition 1 The k out of n ($1 \le k \le n$) Threshold Multiple Confirmer Signatures.

Let $Sign_{DCS}(S, C, m)$ be a general designated confirmer signature (e.g. [10]) on message m, which is signed by S and can be confirmed by C. Assuming there is a group of confirmers $\mathcal{G}_C = \{C_i\}_{i=1,2,\ldots,n}$. We say that $Sign_{TDCS}(S, \mathcal{G}_C, k, m)$ is a k out of n (or (k, n) for short) threshold multiple confirmer signature if any k confirmers within $\mathcal{G}_C$ are capable of helping the verifiers confirm the validity of the signature.

The two special cases on $k = 1$ (i.e., $(1, n)$ scheme) and $k = n$ (i.e., (n, n) scheme) represent any one confirmer, and all confirmers in $\mathcal{G}_C$, are needed for the confirmations.

Definition 2 The $dual\{(1, n), (n, n)\}$ Multiple Confirmer Signatures.

Assuming there is a group of verifiers Λ pre-determined by the signer S. A signature $Sign_{DualDCS}(S, \mathcal{G}_C, \Lambda, m)$ is called the $dual\{(1, n), (n, n)\}$ multiple confirmer signature if the verifier $V_i \in \Lambda$ needs only one confirmer in $\mathcal{G}_C$ to help him confirm the signature, and the verifier not in Λ needs all confirmers in $\mathcal{G}_C$ to complete the confirmation.

2.2 Used Techniques

This section gives some informal definitions and techniques used in our schemes.

Definition 3 Trap-Door Commitment (also see [2, 27]).

Let c be a function with input (y, u, v). The notation y denotes that the public key of the user whose corresponding secret key is x, u is a value committed to and v is a random number. 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 v_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 is proposed in [2, 27].

Trap-door Commitment Example

Let p and q be two large primes and $q \mid p - 1$. 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 the following:

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

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

Multiple Recipients Trap-door Commitment

Jakobsson *et al*. [27] proposed an efficient trap-door commitment scheme for multiple recipients P_i, $i = 1, 2, \ldots, n$. They modified the commitment of u to be $c = g^u \left(\prod_{i=1}^{n} y_i\right)^v \bmod p$. Each recipient is convinced by the proof that u can not be forged by others as long as he knows that his secret key has not been compromised. Any other user would not be convinced by the proof because all P_i, $i = 1, \ldots, n$ could collude to cheat him.

Definition 4 Interactive Bi-proof of Equality (also see [18, 29]).

Fujioka *et al.* in 1992 proposed an interactive bi-proof system that either proved $\log_\alpha(y) = \log_\beta(z)$ or proved $\log_\alpha(y) \neq \log_\beta(z)$. This proof system can be used in our scheme for the confirmer to prove the correctness of the signature to the pre-determined verifiers. 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 $k, \overline{k}, w \in Z_q$, computes $r_\alpha = \alpha^k$, $r_\beta = \beta^k$, $\overline{r}_\alpha = \alpha^{\overline{k}}$ and $\overline{r}_\beta = \beta^{\overline{k}}$, and sends $r_\alpha, r_\beta, \overline{r}_\alpha, \overline{r}_\beta$, 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 $\overline{s} = \overline{k} - (v + w)k \bmod q$, and sends $s, \overline{s}$ to the verifier.
5. The verifier first checks whether $\alpha^s y^{v+w} = r_\alpha$, $\alpha^{\overline{s}} r_\alpha^{v+w} = \overline{r}_\alpha$ and $\beta^{\overline{s}} r_\beta^{v+w} = \overline{r}_\beta$, then he verifies:

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

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

Definition 5 Verifiable Secret Sharing (VSS) (also see [16, 32, 36]).

A VSS scheme is a secret sharing scheme that provides an additional interactive/ noninteractive *Verify* algorithm for each participant in order to verify the validity of his share without communicating with other participants. All participants can recover the same value $\hat{s}$ if their shares are valid. The recovered value $\hat{s}$ is equal to the true secret s if the dealer (responsible for distributing the shares to all participants) is honest.

Pedersen in 1991 proposed a verifiable secret sharing scheme using discrete logarithms [32]. In their scheme, the secret s is given as a discrete logarithm of a public value g^s, and each participant can verify his share using public information in regard to the secret. We used Pedersen's scheme to develop a *Verifiable Discrete Logarithm Shares Construction Proof* (VDLSCP) detailed in section 3.3.

3. PRACTICAL CONSTRUCTIONS

In this section, we propose several practical constructions for multiple designated confirmer signatures. First, we give a general description of $(1, n)$ and (n, n) schemes similar to the schemes proposed by [10] and [29]. A *dual*$\{(1, n), (n, n)\}$ scheme is then presented to subtly combine the schemes of $(1, n)$ and (n, n) multiple confirmer signatures. The *dual*$\{(1, n), (n, n)\}$ scheme provides a variable control over reliability and security, however, we also need to create a (k, n) threshold designated confirmer signature scheme for general applications.

3.1 The $(1, n)$ and (n, n) Multiple Confirmer Signature Schemes

Here, we explain briefly how the Schnorr-like signature [35] can be used to con-

struct the (1, n) and (n, n) multiple confirmer signature schemes. Previously, Okamoto [31] proposed a designated confirmer signature scheme using Schnorr's signature, but there were security flaws presented in [29]. These security flaws have been improved in the following schemes.

Let F be a collision-resistant hash function. We describe the several definitions used in our schemes as follows.

Definition 6 Message-dependent Proof of Equality of the Discrete Logarithm [33].

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 $(w, z) = Proof_{LogEQ}(m, g_1, y_1, g_2, y_2)$, 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: $\log_{g_1}(y_1) \equiv \log_{g_2}(y_2)$. 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 7 Extended Message-dependent Proof of Equality of the Discrete Logarithm.

An extended message-dependent proof of equality of the discrete logarithm of y_i to the base g_i for $i = 1, 2, \ldots, n$ is a tuple $(w, z) = Proof_{ExtLogEQ}(m, g_1, y_1, g_2, y_2, \ldots, g_n, y_n)$, where $w = F(m \| g_1 \| y_1 \| g_2 \| y_2 \| \ldots \| g_n \| y_n \| g_1^z y_1^w \| g_2^z y_2^w \| \ldots \| g_n^z y_n^w)$.

This proof shows that the prover knows the discrete logarithm x: $\log_{g_1}(y_1) \equiv \log_{g_2}(y_2) \ldots \equiv \log_{g_n}(y_n)$. To construct this proof, the prover randomly selects $\kappa \in Z_q$ and calculates $w = F(m \| g_1 \| y_1 \| g_2 \| y_2 \| \ldots \| g_n \| y_n \| g_1^\kappa \| g_2^\kappa \| \ldots \| g_n^\kappa)$ and $z = \kappa - xw \bmod q$.

The (1, *n*) Scheme. We slightly modified the *hinging method* described in [10, 31] and added the $Proof_{ExtLogEQ}$ technique into this scheme, so that any individual confirmer's proof of equality of the discrete logarithm could represent all confirmers' proofs. That is, only one confirmer is needed to verify the validity of a signature even if the capability of confirmation was delegated to n confirmers. Let the notations of p, q, g be defined as in section 2. There are n confirmers $C_1, C_2, \ldots, C_n$ designated by the signer S. The private/public key pairs of the signer S, recipient R, verifier V and the confirmers C_i's are $(x_S, y_S = g^{x_S} \bmod p)$, $(x_R, y_R = g^{x_R} \bmod p)$, $(x_V, y_V = g^{x_V} \bmod p)$ and $(x_{C_i}, y_{C_i} = g^{x_{C_i}} \bmod p)_{i=1,2,\ldots,n}$ respectively. The signing and confirmation protocols are described as follows:

- **Signing Protocol.** The signer S randomly selects $t \in Z_q$ and computes $a = g^t \bmod p$ and $\beta = \prod_{i=1}^{n} b_i$, where $b_i = y_{C_i}^t \bmod p$. S then creates a Schnorr-like signature (e, δ) related to the message m and C_i's public key such that $\rho = g^r \bmod p$ (r is randomly selected by S), $e = F(m \| \rho \| \beta) \oplus a$ (F is a collision resistant hash function), and $\delta = r + e \cdot x_S \bmod q$. Additionally, the signer needs to create $(w_S, z_S) = Proof_{ExtLogEQ}(m, y_{C_1}, b_1, y_{C_2}, b_2, \ldots, y_{C_n}, b_n)$ described in Definition 7 to prove the equality of the discrete logarithm for multiple items: $\log_{y_{C_1}}(b_1) \equiv \log_{y_{C_2}}(b_2) \ldots \equiv \log_{y_{Cn}}(b_n)$. The complete (1, n) multiple designated confirmer signature of a message m is (B, e, δ, w_S, z_S) where $B = (b_1, b_2, \ldots, b_n)$.

- **Proof by the Signer.** The signer S can convince the recipient R that any one confirmer C_i can help R confirm the signature. First, R computes

$$\hat{a} = e \oplus F(m \,\|\, g^{\delta}(y_S^e)^{-1} \| \beta),$$

and then asks the signer S to run the interactive protocol of bi-proof $BP(g, \hat{a}, y_{C_1}, b_1)$ for showing $\log_g(\hat{a}) \equiv \log_{y_{C_1}}(b_1)$. The recipient R also has to check whether the following equation $w_S \stackrel{?}{=} F(m \,\|\, y_{C_1} \,\|\, b_1 \,\|\, y_{C_2} \,\|\, b_2 \,\|\, \ldots \,\|\, y_{C_n} \,\|\, b_n \,\|\, y_{C_1}^{z_S} b_1^{w_S} \,\|\, y_{C_2}^{z_S} b_2^{w_S} \,\|\, \ldots \,\|\, y_{C_n}^{z_S} b_n^{w_S})$ is valid or not. The success of the verification means that R is convinced that $\log_g(\hat{a}) \equiv \log_{y_{C_1}}(b_1) \equiv \log_{y_{C_2}}(b_2) \ldots \equiv \log_{y_{C_n}}(b_n)$ (equals t).
- **Confirmation Protocol.** The verifier V can compute $\hat{a}$ from the signature (B, e, δ, w_S, z_S) and verify whether (w_S, z_S) is created properly (see Proof by the Signer). Any confirmer C_i can run the interactive protocol of bi-proof $BP(g, y_{C_i}, \hat{a}, b_i)$ with V to show $log_g(y_{C_i}) \equiv \log_{\hat{a}}(b_i)$ (equals x_{C_i}). The verifier V could be convinced that the signature is valid if he accepts the proof $BP(g, y_{C_i}, \hat{a}, b_i)$. In this case, an individual confirmer's proof of the discrete logarithm can guarantee the validity of the signature.
- **Conversion Protocol.** Any confirmer C_i can convert the multiple designated confirmer signature to a self-authenticated signature. That is, the verifier V no longer need to ask any C_i to help him verify the signature. Here, C_i randomly selects $\sigma_i \in Z_q$ and computes $\lambda_i = a^{\sigma_i} \bmod p$ and $T_i = \sigma_i + x_{C_i} F(a, \lambda_i) \bmod q$, where F also is a hash function. The confirmer sends (λ_i, T_i) to V, thus, V can verify $a^{T_i} \stackrel{?}{=} \lambda_i b_i^{F(a,\lambda_i)}$ [10].

The (n, n) Scheme. The (n, n) scheme can be constructed by modifying the $(1, n)$ scheme into an easier process. Let $Y_C = \prod_{i=1}^{n} y_{C_i}$ is a shared public key for the con firmers $C_1, C_2, \ldots, C_n$ and $\beta = \prod_{i=1}^{n} b_i = Y_C^t$. The signature of (n, n) scheme is represented by (β, e, δ), where e and δ are the same as in the $(1, n)$ scheme. In the Proof by the Signer phase, the recipient R computes $\hat{a} = e \oplus F(m \,\|\, g^{\delta}(y_S^e)^{-1} \,\|\, \beta)$ and asks the signer S to run the interactive protocol of bi-proof $BP(g, \hat{a}, Y_C, \beta)$ to show $\log_g(\hat{a}) \equiv \log_{Y_C}(\beta)$. Similarly, in the Confirmation phase, all confirmers $\{C_i\}_{i=1,2,\ldots,n}$ cooperatively run the *bi-proof for multiple provers* $BP_{\{C_1,\ldots,C_n\}}(g, Y_C, \hat{a}, \beta)$ with the verifier V to show $\log_g(Y_C) \equiv \log_{\hat{a}}(\beta)$ (equals $\sum_{i=1}^{n} x_{C_i}$). The bi-proof for multiple provers can easily be constructed from the original bi-proof protocol, and so we omitted the details here.

3.2 The *dual*{(1, *n*), (*n*, *n*)} Multiple Confirmer Signature Scheme

This section presents a scheme about the additional control over specified verifiers $V_i \in \Lambda$ designated by the signer. The scheme can be regarded as a $(1, n)$-type scheme for these $V_i \in \Lambda$, but on the contrary, the scheme can only be a (n, n)-type scheme for the other verifiers $(\notin \Lambda)$. The control factor c would be added into the extended message-dependent proof of equality of the discrete logarithm in determining who can obtain the conviction of the correctness of the proof.

Definition 8 Designated Verifier Extended 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 extended the message-dependent proof of equality of the discrete logarithm of y_i to the base g_i for $i = 1, 2, \ldots, n$ is a tuple $(w, z, u, v) = Proof_{DVExtLogEQ}(m, c, y_V, g_1, y_1, g_2, y_2, \ldots, g_n, y_n)$, where $w = F(m \| c \| g_1 \| y_1 \| g_2 \| y_2 \| \ldots \| g_n \| y_n \| g_1^z y_1^{(w+u)} \| g_2^z y_2^{(w+u)} \| \ldots \| g_n^z y_n^{(w+u)})$ and $c = g^u y_V^v \bmod p$ is a trap-door commitment.

The prover, using this proof, can convince the designated verifier V that he knows the discrete logarithm x: $\log_{g_1}(y_1) \equiv \log_{g_2}(y_2) \ldots \equiv \log_{g_n}(y_n)$. To construct this proof, the prover randomly selects $u, v, \kappa \in Z_q$ and calculates $c = g^u y_V^v \bmod p$, $w = F(m \| c \| g_1 \| y_1 \| g_2 \| y_2 \| \ldots \| g_n \| y_n \| g_1^{\kappa} \| g_2^{\kappa} \| \ldots \| g_n^{\kappa})$ and $z = \kappa - x(w + u) \bmod q$.

According to the Definition 8, a $dual\{(1, n), (n, n)\}$ scheme for a single verifier can be represented by $(B, e, \delta, w_S, z_S, u_S, v_S)$. Only the specified verifier V can be convinced by $Proof_{DVExtLogEQ}(m, c, y_V, y_{C_1}, b_1, y_{C_2}, b_2, \ldots, y_{C_n}, b_n)$, that is, can believe that

$$\log_{y_{C_1}}(b_1) \equiv \log_{y_{C_2}}(b_2) \ldots \equiv \log_{y_{C_n}}(b_n).$$

Thus, as long as any one confirmer C_i proves that $\log_g(\hat{a}) \equiv \log_{y_{Ci}}(b_i)$, V can obtain the conviction of the validity of the signature. However, other verifiers who cannot be convinced by $Proof_{DVExtLogEQ}$ must ask all confirmers to cooperatively illustrate the relationship between $\hat{a}$ and $\prod_{i=1}^{n} b_i$ (refer to the (n, n) scheme).

To achieve the goal of multiple designated verifiers in a $dual\{(1, n), (n, n)\}$ scheme, we modified c to be $c = g^u \left(\prod_{i=1}^{s_d} g_{V_i} \right)^v \bmod p$. Then, the verifiers $v_1, V_2, \ldots, V_{s_d}$ can be convinced that the signature is valid even if only one confirmer proves the confirmation protocol to them.

3.3 The (k, n) Threshold Multiple Confirmer Signature Scheme

To construct a (k, n) threshold multiple confirmer signature scheme, a *Verifiable Discrete Logarithm Shares Construction Proof* (VDLSCP) must be developed to help the verifier check if the signature is constructed properly or not. Let $f(x) = d_0 + d_1 x + d_2 x^2 + \ldots + d_{k-1} x^{k-1}$ be a random polynomial over Z_q with degree $k - 1$. The signer publishes $a_i = g^{f(x_i)} \bmod p$, for $i = 1, 2, \ldots, n$, and $g^{d_i} \bmod p$, for $i = 0, 1, 2, \ldots, k - 1$, the verifier therefore can be convinced that the discrete logarithm of a_i is a correct share of the secret $f(0) = d_0$ by running VDLSCP.

Definition 9 Verifiable Discrete Logarithm Shares Construction Proof (VDLSCP).

There are two algorithms of VDLSCP: $VDLSCP_{Construct}(D, f(x), \{x_i\}_{i \in [1,n]})$ and $VDLSCP_{Verify}(V, \{g^{d_i} \bmod p\}_{i \in [0,k-1]}, \{a_i\}_{i \in [1,n]})$. These two algorithms are similar to Pedersen's VSS scheme. In Pedersen's VSS, the dealer privately distributes the secret shares $f(x_i)$'s to the participants, but in our VDLSCP, the dealer publicly sends $a_i = g^{f(x_i)}$, $i = 1, 2, \ldots, n$ to the verifier instead.

• $VDLSCP_{Construct}(D, f(x), \{x_i\}_{i \in [1,n]})$ algorithm:

1. The dealer D randomly chooses a degree $k-1$ polynomial $f(x) = d_0 + d_1x + \ldots + d_{k-1}x^{k-1}$ over Z_q where $f(0) = d_0$.
2. The dealer computes $f(x_i)$ for $i \in [1, n]$ where all x_i's are public values.
3. The dealer publicly sends $a = g^{f(0)} \bmod p$, $g^{d_i} \bmod p$, for $i = 1, 2, \ldots, k-1$ and $a_i = g^{f(x_i)} \bmod p$, for $i = 1, 2, \ldots, n$, to the verifier.

- $VDLSCP_{Verify}(V, a, \{g^{d_i} \bmod p\}_{i\in[0,k-1]}, \{a_i\}_{i\in[1,n]})$ algorithm:
  1. The verifier V computes $h_i = a \cdot \prod_{j=1}^{k-1} (g^{d_j})^{x_i^j} \bmod p$, for all $i = 1, 2, \ldots, n$.
  2. The verifier V verifies whether $h_i \stackrel{?}{=} a_i$. If it holds, then he accepts that a_i is constructed properly, i.e., the discrete logarithm of a_i is a correct share of the secret $f(0) = d_0$. This means that the discrete logarithms of any k a_i's (in all $\{a_i\}_{i=1,2,\ldots,n}$) can be used to recover the same secret value.

We describe the (k, n) threshold multiple confirmer signature scheme in the following:

- **Signing Protocol.** The signer runs the algorithm of $VDLSCP_{Construct}(S, f(x), \{y_{C_i}\}_{i\in[1,n]})$ to construct the public values related to the secret shares, and uses these public values to generate the confirmer signature. That is, the signer S randomly selects a secret polynomial $f(x) = d_0 + d_1x + d_2x^2 + \ldots + d_{k-1}x^{k-1}$ and computes $t_i = f(y_{C_i})$, for $i = 1, 2, \ldots, n$. The signer then computes $a = g^{f(0)} \bmod p$, $a_i = g^{t_i} \bmod p$, for $i \in [1, n]$ and $g^{d_i} \bmod p$, for $i \in [1, k-1]$. Afterwards, the signer S computes $b_i = y_{C_i}^{t_i} \bmod p$ and creates a Schnorr-like signature (e, δ) related to the message m and C_i's public key such that $\rho = g^r \bmod p$ (r is randomly selected by S), $e = F(m \| \rho \| \cup_b) \oplus a$, and $\delta = r + e \cdot x_S \bmod q$, where $\cup_b = (b_1 \| b_2 \| \ldots \| b_n)$. The complete (k, n) multiple designated confirmer signature of a message m is (A, B, L, e, δ) where $A = (a_1, a_2, \ldots, a_n)$, $B = (b_1, b_2, \ldots, b_n)$ and $L = (g^{d_1}, g^{d_2}, \ldots, g^{d_{k-1}})$.
- **Proof by the Signer.** The signer S can convince the recipient R that any k out of n confirmers can help R confirm the signature. First, R computes

$$\hat{a} = e \oplus F(m \| g^{\delta}(y_S^e)^{-1} \| \cup_b),$$

  and then runs the algorithm of $VDLSCP_{verify}(R, \hat{a}, L, A)$ to verify if each a_i is constructed correctly or not. Aside from this, R must ask the signer S to run the interactive protocol of bi-proof $BP(g_i, a_i, y_{C_i}, b_i)$, for all $i \in [1, n]$, to show $\log_g(a_i) \equiv \log_{y_{C_i}}(b_i)$.
- **Confirmation Protocol.** The verifier V can compute $\hat{a}$ from the signature (A, B, L, e, δ) and run the algorithm of $VDLSCP_{verify}(R, \hat{a}, L, A)$ to verify each a_i (see Proof by the Signer). Any group of k confirmers (for simplicity, assume that $C_1, C_2, \ldots, C_k$) can run the interactive protocol of bi-proof $BP(g, y_{C_i}, a_i, b_i)$, for $i \in [1, k]$ with V to show that $\log_g(y_{C_i}) \equiv \log_{a_i}(b_i)$ (equals x_{C_i}).
- **Conversion Protocol.** The k confirmers $C_1, C_2, \ldots, C_k$ can convert the (k, n) threshold designated confirmer signature to a general signature. This means that the verifier V no longer need to ask the confirmers to help him verify the signature. Here, each C_i, $i \in [1, k]$ randomly selects $\sigma_i \in Z_q$ and computes $\lambda_i = a_i^{\sigma_i} \bmod p$ and $T_i = \sigma_i + x_{C_i}F(a_i, \lambda_i) \bmod q$, where F also is a hash function. The confirmer sends (λ_i, T_i) to V, thus, V can verify $a_i^{T_i} \stackrel{?}{=} \lambda_i b_i^{F(a_i,\lambda_i)}$.

4. SECURITY ANALYSIS

In this section, we analyze the security of our scheme. We assume that in the random oracle model [34], the Schnorr signature scheme and the Extended Message-dependent Proof of Equality of the Discrete Logarithm can be proven to be secure.

4.1 The Security of the (1, *n*) and (*n*, *n*) Scheme

The following security properties are discussed in the (1, *n*) scheme. The security analysis to the (*n*, *n*) scheme is very similar and is omitted here.

Unforgeability of Signatures. Two scenarios of forging the confirmer signatures are described here to analyze our scheme's security. Assuming a forging algorithm $\mathcal{A}_1$ creates a new value $e' \neq e$ and computes $\delta' = r' + e'x_S$. In this scenario, $\mathcal{A}_1$ can select a random t' and calculate $a' = g^{t'} \bmod p$ and $b_i' = y_{C_i}^{t'} \bmod p$. Then $\mathcal{A}_1$ calculates $\beta' = \prod_{i=1}^{n} b_i'$, $\rho' = g^{r'} \bmod p$ and $e' = F(m^* \| \rho' \| \beta') \oplus a'$. This forged signature (e', δ') can successfully pass the verification of bi-proof of both signer and confirmer. However, it is difficult to find a valid δ' without knowing the signer's secret x_S. Thus, forging a correct confirmer signatures using $\mathcal{A}_1$ is as difficult as breaking Shnorr's signature.

In another scenario, a forging algorithm $\mathcal{A}_2$ tries to find a value of $e^+ = F(m^* \| \rho \| \beta^+) \oplus a^+ = e = F(m \| \rho \| \beta) \oplus a$. In this scenario, $\mathcal{A}_2$ need not find a new value of δ to forge a valid signature. The two methods in determining β^+ and a^+ which can satisfy $F(m^* \| \rho \| \beta^+) \oplus a^+ = e$ are described as follows: (1) $\mathcal{A}_2$ randomly selects b_i^+ and computes $\beta^+ = \prod_{i=1}^{n} b_i^+$ and $a^+ = F(m^* \| \rho \| \beta^+) \oplus e$. $\mathcal{A}_2$ can easily obtain a^+; however, the discrete logarithms of a^+ and β^+ are not the same because F is a collision-resistant hash function, whose output is truly random. (2) $\mathcal{A}_2$ randomly selects a^+, and computes β^+ that can satisfy $F(m^* \| \rho \| \beta^+) \oplus a^+ = e$. However, since F is a one-way hash function, to find a proper β^+ is computationally infeasible.

Indistinguishhability of Signatures. The following lemma is used to analyze indistinguishability in our scheme.

Lemma 1 Decision-Diffie-Hellman Assumption [29].

Let two sets be defined as follows:

$\chi = \{(g1, g2, y1, y2) \in G^4 \mid \langle g1 \rangle = \langle g2 \rangle = G\}$
$\mathcal{DH} = \{(g1, g2, y1, y2) \in \chi \mid \log_{g_1} y_1 = \log_{g_2} y_2$.

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 a simulator $\mathcal{A}$ that can distinguish a valid confirmer signature from a simulated one. We show that we can use $\mathcal{A}$ to solve the Decision-Diffie-

Hellman problem (Lemma 1). According to Lemma 1, assume that $\mathcal{A}$ generates two simulated signatures $\mathcal{S}_1 = (B^*, e^*, \delta^*, w_S^*, z_S^*)$ and $\mathcal{S}_2 = (B^+, e^+, \delta^+, w_S^+, z_S^+)$, where $\mathcal{S}_1 \in \mathcal{DH}$ is a valid signature and $\mathcal{S}_2 \in \chi$ is an invalid one. If $\mathcal{A}$ can identify the correct signature from $\mathcal{S}_1$ and $\mathcal{S}_2$, we can tell which pair of (a^*, β^*) or (a^+, β^+) has the same discrete logarithm, i.e. comes from $\mathcal{DH}$. This result violates the assumption of Lemma 1.

4.2 The Security of the *dual*{(1, *n*), (*n*, *n*)} Scheme

The security of the *dual*{(1, *n*), (*n*, *n*)} scheme relies on Definition 8. A designated verifier V who knows the secret x_V can be convinced that $Proof_{DVExtLogEQ}$ is correct, while others cannot be convinced. Below, we will show that V can simulate the correct transcripts of $Proof_{DVExtLogEQ}$ to cheat anyone even if V doesn't know the prover's secret.

Lemma 2 Simulation of the Proof.

Let V be a designated verifier who has a secret/public key pair $(x_V, y_V = g^{x_V} \bmod p)$. Without accessing the prover's secret, V is able to forge $Proof_{DVExtLogEQ}$ by randomly selecting τ, γ, and $z \in Z_q$ and calculating:

$c = g^{\tau} \bmod p$
$w = F(m \| c \| g_1 \| y_1 \| g_2 \| y_2 \| \ldots \| g_n \| y_n \| g_1^z y_1^{\gamma} \| g_2^z y_2^{\gamma} \| \ldots \| g_n^z y_n^{\gamma})$
$u = (\gamma - w) \bmod q$
$v = (\tau - u) x_V^{-1} \bmod q$.

The (w, z, u, v) result will then successfully pass the proof verification.

4.3 The Security of (*k*, *n*) Threshold Scheme

If the signer does not trust a single confirmer, he may want to delegate the confirmation ability to n agents such that the verification will require at least k number of these to cooperate. However, a threshold (k, n) scheme must prevent $k - 1$ or less confirmers from colluding to convince V into believing the correctness of the signature. The following lemma will show that this type of collusion cannot be successfully performed in our scheme.

Lemma 3 The Collusion Attack on the Threshold Confirmer Signature Scheme.

In our (k, n) threshold scheme, $(k - 1)$ or less confirmers cannot collude to convince the verifier V that a valid signature is correct even if they leak their secrets to each other.

We prove this lemma by contradiction. We show that anyone can create an invalid signature that passes the confirmation protocol performed by only $k - 1$ confirmers. According to the confirmation protocol, we assume that $C_1, C_2, \ldots, C_{k-1}$ run the interactive protocol of bi-proof to show $\log_g(y_{Ci}) \equiv \log_{a_i}(b_i)$, for $i \in [1, k - 1]$. A malicious user who obtained a valid signature (A, B, L, e, δ) for message m can take the following steps to create an invalid signature for m^* by passing the confirmation protocols performed by these $k - 1$ confirmers.

1. Computes $a^* = F(m^* \| \rho \| \cup_b) \oplus e$.
2. Let $\{(px_0, g^{py_0}), (px_1, g^{py_1}), \ldots, (px_{k-1}, g^{py_{k-1}})\} = \{(0, a^*), (y_{C_1}, a_1), \ldots, (y_{C_{k-1}}, a_{k-1})\}$. Constructs a polynomial over the base g:

$$g^{f(x)^*} = \prod_{s=0}^{k-1} \left(g^{py_s}\right)^{\prod_{j=0, j\neq s}^{k-1} \frac{x-px_j}{px_s-px_j}} = g^{d_0^*} \cdot (g^{d_1^*})^x \ldots (g^{d_{k-1}^*})^{x^{k-1}}.$$

3. Computes the values of $(a_k^*, a_{k+1}^*, \ldots, a_n^*) = (g^{f(y_{C_k})^*}, g^{f(y_{C_{k+1}})^*}, \ldots, g^{f(y_{C_n})^*})$.
4. Create an invalid signature (A^*, B, L^*, e, δ), where $A^* = (a_1, a_2, \ldots, a_{k-1}, a_k^*, a_{k+1}^*, a_n^*)$ and $L^* = (g^{d_1^*}, g^{d_2^*}, \ldots, g^{d_{k-1}^*})$.

Unforgeability of Signatures. A (k, n) threshold multiple confirmer signature is valid if it passes the confirmation protocol performed by k or more confirmers. To prove this, we created two scenarios in which a signature is forged. In the forging algorithm $\mathcal{A}_1$, a new value $e' \neq e$ needs to be computed. $\mathcal{A}_1$ randomly selects a polynomial $f(x)' = d_0' + d_1'x + d_2'x^2 + \ldots + d_{k-1}'x^{k-1}$ and computes $t_i' = f(y_{C_i})'$, for $i = 1, 2, \ldots, n$. The signer then computes $a' = g^{f(0)'} \bmod p$, $a_i = g^{t_i'} \bmod p$, $b_i = y_{C_i}^{t_i'} \bmod p$ $(i \in [1, n])$ and $g^{d_i'} \bmod p$ $(i \in [1, k-1])$. Thus, e' can be computed as $e' = F(m^* \| \rho' \| \cup_b') \oplus a'$, where $\cup_b' = (b_1' \| b_2' \| \ldots \| b_n')$. However, without knowing the signer's secret x_S, a Schnorr-like signature $\delta' = r' + e'x_S$ cannot be obtained.

In another scenario, a forging algorithm $\mathcal{A}_2$ tries to find a value for $e^+ = F(m^* \| \rho \| \cup_b^+) \oplus a^+ = e = F(m \| \rho \| \cup_b) \oplus a$. Here, $\mathcal{A}_2$ doesn't need to find a new value of δ to forge a valid signature. The two methods in determining B^+ and A^+ that can satisfy $F(m^* \| \rho \| \cup_b^+) \oplus a^+ = e$ will be described as follows: (1) $\mathcal{A}_2$ randomly selects b_i^+ and computes $a^+ = F(m^* \| \rho \| \cup_b^+) \oplus e$, where $\cup_b^+ = (b_1^+ \| b_2^+ \| \ldots \| b_n^+)$. $\mathcal{A}_2$ can easily obtain a^+; however, a^+ has to be a random number because F is a collision-resistant hash function, whose output is truly random. Since all b_i's have been previously determined and at least k confirmers have proven their confirmer protocols, we know that at least k a_i's must be fixed according to b_i's. The critical problem here is that these k a_i's can also determine the value of a^+ using the Lagrange interpolating method in finding a unique polynomial. That means that the probability of finding a proper a^+ to satisfy the above two conditions is negligible. (2) $\mathcal{A}_2$ randomly selects a^+ and computes $\cup_b^+$ that can satisfy $F(m^* \| \rho \| \cup_b^+) \oplus a^+ = e$. However, since F is a one-way hash function, finding a proper $\cup_b^+$ is computationally infeasible.

5. THE APPLICATION ON THE FAIR EXCHANGE PROTOCOL

The protocol of fair exchange of digital signatures [1, 4, 13], which involves two signing party exchange their signatures in a fair manner, is an important application for the designated confirmer signature. An efficient strategy in maintaining the fairness of the protocol can be done using an off-line trusted third party (off-line TTP) as a mediator to solve disputes between the two attending parties. However, the above strategy depends on a fully trustworthy center, which is a high-risk assumption. Any center may suffer the attacks of viruses or Trojan horses, or may be compromised by a malicious intruder.

Franklin and Reiter [17] have proposed a semi-trusted third party (STTP) model that can improve the security by protecting the exchanging secrets against the misuse of the STTP.

Nevertheless, the model of STTP cannot solve the problems of colluding attacks and system reliability. One of the two attending parties may collude with the STTP to gain an advantage over the other party. Aside from this, the service may be stopped if the STTP is unavailable due to being faulty or compromised. Our proposed multiple confirmer signature scheme can be applied to solve the above problems by adjusting the level of trust and the degree of reliability. Below, we use the (k, n) threshold scheme as an example to describe a highly reliable protocol for fair exchange of signatures with multiple off-line TTPs.

The protocol involves n + 2 parties, namely the two attending parties U_A and U_B, and n off-line TTPs named T_1, T_2, …, T_n. Assuming that U_A and U_B want to exchange their signatures on the same message m, the protocol is divided into two phases:

Normal Phase

1. U_A signs a signature s_A on the message m by using (k, n) threshold multiple confirmer signature scheme and sends s_A to U_B.
2. U_A convinces U_B that s_A is a valid signature and can be confirmed and converted by k or more TTPs out of $\{T_1, T_2, \ldots, T_n\}$.
3. U_B sends his universally verifiable signature Sig_B on the message m to U_A. Note that Sig_B can be made by any original signature scheme such as RSA or DSS, and can be publicly verified by everyone.
4. U_A verifies Sig_B. If the verification succeeds, U_A also sends his universally verifiable signature Sig_A on the message m to U_B.

Dispute Phase

If U_B does not receive U_A's signature Sig_A after he has sent Sig_B to U_A, he can ask TTPs to solve the dispute.

1. U_B sends s_A to all n TTPs, T_1, T_2, …, T_n, for conversion. In addition, U_B also needs to send Sig_B to all TTPs (a more complex and secure version: U_B also needs to create a (k, n) threshold scheme for Sig_B to divide Sig_B into n pieces and sends each piece Sig_{Bi} to T_i, for i = 1, 2, …, n).
2. Each T_i checks s_A and Sig_B (or Sig_{Bi}). If the verification succeeds, T_i sends the conversion message to U_B and sends Sig_B (or Sig_{Bi}) to U_A.
3. If U_B receives the conversion messages from k or more out of n TTPs, U_B can then convert s_A into a self-authenticated (universally verifiable) signature.

Lemma 4 Fairness property.

If k or more TTPs are honest, the above protocol (more complex version) can guarantee that, either both parties U_A and U_B gain each other's signature or they obtain nothing valid (note that k must be greater than or equal to $\left\lfloor \frac{n}{2} \right\rfloor + 1$ if dishonest TTPs may conspire with U_A or U_B).

The signature s_A in the above protocol can also be constructed by using the (n, n) or $dual\{(1, n), (n, n)\}$ scheme according to different security and reliability policies. Aside from this, we only give a brief description for the application of our proposed multiple confirmer signature scheme. The above basic model also can be further extended to a more advanced version such as ensuring a timely termination [1] and adding an abuse-free property [20].

6. COMPARISONS

In this section, we compare our schemes with Michels and Stadler's schemes [29]. In Michels and Stadler's paper, only the $(1, n)$ scheme was described in detail (the (n, n) scheme can be easily constructed by extending the $(1, n)$ scheme, but there is no specific procedure presented in their paper). Thus, we compare the efficiency of the $(1, n)$ scheme in our paper and Michels and Stadler's paper. Table 1 shows the comparison results. Aside from this, an overall difference between our paper and Michels and Stadler's paper is shown in Table 2.

Table 1. Comparison of efficiency.

	Michels and Stadler's scheme [29] (Schnorr-like)	Our proposed scheme
$(1, n)$ scheme	Efficiency: Signing: $(n + 2)$ *exp* + 2 *hash* + 2 *add*(G_q) + 1 *mul*(G_q) Proof by the Signer: 3 *exp* + 2 *hash* + *n mul*(G_q) + 1 *bi-proof-for-n-discrete-logs* Confirmation: 3 *exp* + 2 *hash* + 1 *mul*(G_q) + 1 *bi-proof* Conversion: not mentioned	Efficiency: Signing: $(n + 2)$ *exp* + 1 *hash* + 1 *add*(Z_p^*) + 1 *add*(G_q) + $n - 1$ *mul*(Z_p^*) + 1 *mul*(G_q) Proof by the Signer: 2 *exp* + 1 *hash* + $(n - 1)$ *mul*(Z_p^*) + 1 *add*(Z_p^*) + 1 *bi-proof* + 1 $Proof_{ExtLogEQ}$ Confirmation: 2 *exp* + 1 *hash* + $(n - 1)$ *mul*(Z_p^*) + 1 *add*(Z_p^*) + 1 *bi-proof* + 1 *Verify-Proof*$_{ExtLogEQ}$ Conversion: 2 *exp* + 2 *hash* + 1 *add*(G_q) + 1 *mul*(Z_p^*) + 1 *mul*(G_q)

Table 2. Overall difference.

	Michels and Stadler's scheme [29]	Our proposed scheme
$(1, n)$ scheme	Use of the bi-proof to prove the equality of n discrete logarithms	Use of a non-interactive proof, $Proof_{ExtLogEQ}$, to prove the equality of n discrete logarithms
(n, n) scheme	No specific procedure	Extension of $(1, n)$ scheme
$dual\{(1, n), (n, n)\}$ scheme	None	Use of a non-interactive proof, $Proof_{DVExtLogEQ}$, to prove the equality of n discrete logarithms to a specified group of verifiers
(k, n) scheme	No specific procedure	Use of the verifiable secret sharing scheme and a special construction of the hinging factors (a, b)

The notation is described as follows. *exp* denotes an exponentiation operation over Z_p^*, *hash* denotes a collision-resistant hash operation, *add* denotes an addition operation and *mul* denotes a multiplication operation. Since the techniques of our scheme and Michels and Stadler's scheme are different, we can only roughly evaluate the numbers of the basic operations. For example, we do not consider the differences in input and output sizes of the hash functions, which are too complicated. For simplification, the detailed operations of some proofs, such as *bi-proof* and $Proof_{ExtLogEQ}$, are also omitted.

The computational cost of our (1, *n*) scheme is very close to that of Michels and Stadler's scheme. However, Michels and Stadler employ the bi-proof to prove that *n* discrete logarithms are the same. Our scheme instead suggests a non-interactive proof $Proof_{ExtLogEQ}$, which is a portion of the signature, to show the equality of *n* discrete logarithms. Thus, anyone who obtains the $Proof_{ExtLogEQ}$ can verify it by himself/herself and thus the confirmation procedure can be done by the aid of only one confirmer. This is the reason that the Confirmation phase of our proposed scheme needs an additional operation of *Verify-*$Proof_{ExtLogEQ}$ (see Table 1). Moreover, a particular and useful extended proof $Proof_{DVExtLogEQ}$, the basic cryptographic primitive of building the *dual*{(1, *n*), (*n*, *n*)} scheme, is systematically developed. The *dual*{(1, *n*), (*n*, *n*)} scheme can thus be implemented by some minor modifications to the (1, *n*) and (*n*, *n*) schemes.

The (*k*, *n*) scheme is an another advantage of our paper. It is particularly useful for enhancing the security of the application system, such as the fair exchange protocol. The fair exchange protocol mentioned in the previous section can reach the same goal of the "semi-trusted" TTP proposed in [17], and can further improve the reliability of the protocols in [17]. To sum it up, Michels and Stadler contributed several generic construction models for building the confirmer signature with many existing signature schemes; however, we are proposing some flexible models which are more suitable in constructing a variety of multiple confirmer signature schemes.

7. CONCLUSIONS

This paper presents several practical schemes for a multiple confirmer signature. A multiple confirmer signature scheme gives advantages to both the signer and recipient, because a signer simply cannot trust a single confirmer and a recipient has the assurance of being able to convince others that the signature is correct. As there are various models in our scheme, the tradeoff between security and reliability can easily be adjusted.

A generalized multiple confirmer signature scheme, like the concept of Generalized Group-oriented Cryptosystem (GGOC) [12], can be realized by extending our basic (*k*, *n*) threshold scheme; however, the details are omitted here. Moreover, some applications such as fair exchange and contract signing can be improved by adopting the framework of our multiple confirmers scheme to maintain a highly reliable and highly secure service.

REFERENCES

1. N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures," *IEEE Journal on Selected Areas in Communications*, Vol. 18, 2000, pp. 593-610.

2. G. Brassard, D. Chaum, and C. Crepeau, "Minimum disclosure proofs of knowledge," *Journal of Computer and System Sciences*, Vol. 37, 1988, pp. 156-189.
3. J. Boyar, D. Chaum, I. Damgard, and T. Pedersen, "Convertible undeniable signatures," *Advances in Cryptology – CRYPTO*, LNCS 537, Springer-Verlag, 1990, pp. 189-205,.
4. F. Bao, R. H. Deng, and W. Mao, "Efficient and practical fair exchange protocols with off-line TTP," in *Proceedings of the IEEE Symposium on Security and Privacy*, 1998, pp. 77-85.
5. C. Boyd and E. Foo, "Off-line fair payment protocols using convertible signatures," *Advances in Cryptology – ASIACRYPT*, LNCS 1514, Springer-Verlag, 1998, pp. 271-285.
6. J. Camenisch and M. Michels, "Confirmer signature schemes secure against adaptive adversaries," *Advances in Cryptology – EUROCRYPT*, LNCS 1807, Springer-Verlag, 2000, pp. 243-258.
7. D. Chaum and H. van Antwerpen, "Undeniable signatures," *Advances in Cryptology – CRYPTO*, LNCS 435, Springer-Verlag, 1989, pp. 212-217.
8. D. Chaum, "Zero-knowledge undeniable signatures," *Advances in Cryptology – EUROCORYPT*, LNCS 473, Springer-Verlag, 1990, pp. 458-464.
9. D. Chaum, "Some weaknesses of weaknesses of undeniable signatures," *Advances in Cryptology – EUROCRYPT*, LNCS 547, Springer-Verlag, 1991, pp. 554-556.
10. D. Chaum, "Designated confirmer signatures," *Advances in Cryptology – EUROCRYPT*, LNCS 950, Springer-Verlag, 1994, pp. 86-91.
11. D. Chaum, E. van Heijst, and B. Pfitzmann, "Cryptographically strong undeniable signatures, unconditionally secure for the signer," *Advances in Cryptology – CRYPTO*, LNCS 576, Springer-Verlag, 1991, pp. 470-484.
12. C. C. Chang and H. C. Lee, "A new generalized group-oriented cryptoscheme without trusted centers," *IEEE Journal on Selected Areas in Communications*, Vol. 11, 1993, pp. 725-729.
13. L. Chen, "Efficient fair exchange with verifiable confirmation of signatures," *Advances in Cryptology – ASIACRYPT*, LNCS 1514, Springer-Verlag, 1998, pp. 286-299.
14. I. Damgard and T. Pedersen, "New covertible undeniable signature schemes," *Advances in Cryptology – EUROCRYPT*, LNCS 1070, Springer-Verlag, 1996, pp. 372-386.
15. Y. Desmedt and M. Yung, "Weaknesses of undeniable signature schemes," *Advances in Cryptology – EUROCRYPT*, LNCS 547, Springer-Verlag, 1991, pp. 205-220.
16. P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *Proceedings of the 28th IEEE Symposium on the Foundations of Computer Science*, 1987, pp. 427-437.
17. M. K. Franklin and M. K. Reiter, "Fair exchange with a semi-trusted third party," in *Proceedings of the 4th ACM Conference on Computer and Communications Security*, 1997, pp. 1-5.
18. A. Fujioka, T. Okamoto, and K. Ohta, "Interactive bi-proof systems and undeniable signature schemes," *Advances in Cryptology – EUROCRYPT*, LNCS 547, Springer-Verlag, 1991, pp. 243-256.
19. S. Galbraith and W. Mao, "Invisibility and anonymity of undeniable and confirmer

signatures," *Topics in Cryptology – CT-RSA*, LNCS 2612, Springer-Verlag, 2003, pp. 80-97.
20. J. A. Garay, M. Jakobsson, and P. MacKenzie, "Abuse-free optimistic contract signing," *Advances in Cryptology – CRYPTO*, LNCS 1666, Springer-Verlag, 1999, pp. 449-466.
21. R. Gennaro, H. Krawczyk, and T. Rabin, "RSA-based undeniable signatures," *Advances in Cryptology – CRYPTO*, LNCS 1294, Springer-Verlag, 1997, pp. 132-149.
22. R. Gennaro, H. Krawczyk, and T. Rabin, "Undeniable certificates," *Electronic Letters*, Vol. 35, 1999, pp. 1723-1724.
23. S. Goldwasser and E. Waisbard, "Efficient transformation of well known signature schemes into designated confirmer signature schemes," Technical Report No. MCS03-13, Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Israel, 2003.
24. L. Gong, "Increasing availability and security of an authentication service," *IEEE Journal on Selected Areas in Communications*, Vol. 11, 1993, pp. 657-662.
25. S. Han, W. K. Y. Yeung, and J. Wang, "Identity-based confirmer signatures from pairings over Elliptic curves," in *Proceedings of the 4th ACM Conference on Electronic Commerce*, 2003, pp. 262-263.
26. M. Jakobsson, "Blackmailing using undeniable signatures," *Advances in Cryptology – EUROCRYPT*, LNCS 950, Springer-Verlag, 1994, pp. 425-427.
27. M. Jakobsson, K. Sako, and R. Impagliazzo, "Designated verifier proofs and their applications," *Advances in Cryptology – EUROCRYPT*, LNCS 1070, Springer-Verlag, 1996, pp. 143-154.
28. M. Michels, "Breaking and repairing a convertible undeniable signature scheme," in *Proceedings of the ACM Conference on Computer and Communications Security*, 1996, pp. 148-152.
29. M. Michels and M. Stadler, "Generic constructions for secure and efficient confirmer signature schemes," *Advances in Cryptology – EUROCRYPT*, LNCS 1403, Springer-Verlag, 1998, pp. 406-421.
30. K. Nguyen, Y. Mu, and V. Varadharajan, "Undeniable confirmer signature," in *Proceedings of the 2nd International Workshop on Information Security*, LNCS 1729, Springer-Verlag, 1999, pp. 235-246.
31. T. Okamoto, "Designated confirmer signatures and public-key encryption are equivalent," *Advances in Cryptology – CRYPTO*, LNCS 839, Springer-Verlag, 1994, pp. 61-74.
32. T. Pedersen, "Distributed provers with applications to undeniable signatures," *Advances in Cryptology – EUROCRYPT*, LNCS 547, Springer-Verlag, 1991, pp. 221-242.
33. H. Petersen, "How to convert any digital signature scheme into a group signature scheme," in *Proceedings of Security Protocols Workshop*, LNCS 1361, Spring-Verlag, 1997, pp. 67-78.
34. D. Pointcheval and J. Stern, "Security proofs for signatures," *Advances in Cryptology – EUROCRYPT*, LNCS 1070, Springer-Verlag, 1996, pp. 387-398.
35. C. P. Schnorr, "Efficient signature generation for smart cards," *Journal of Cryptology*, Vol. 4, 1991, pp. 161-174.
36. M. Stadler, "Publicly verifiable secret sharing," *Advances in Cryptology – EURO-*

CRYPT, LNCS 1070, Springer-Verlag, 1996, pp. 190-199.
37. S. P. Wang, Y. M. Wang, and Y. L. Zhang, "A confirmer signature scheme based on DSA and RSA," *Journal of Software*, Vol. 14, 2003, pp. 588-593.
38. F. Zhang, R. Safavi-Naini, and W. Susilo, "Attack on Han *et al.*'s ID-based confirmer (undeniable) signature at ACM-EC'03," 2003, http://eprint.iacr.org.

Chih-Hung Wang (王智弘) was born in Kaohsiung Taiwan, in 1968. He received the B.S. degree in Information Science from Tunghai University and M.S. degree in Information Engineering from National Chung Cheng University, Taiwan, R.O.C., in 1991 and 1993, respectively. He received the Ph.D. 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.