Designated Confirmer Signatures and Public-Key Encryption are Equivalent

Tatsuaki Okamoto

NTT Laboratories
Nippon Telegraph and Telephone Corporation
1-2356 Take, Yokosuka-shi, Kanagawa-ken, 238-03 Japan
Email: okamoto@sucaba.ntt.jp

Abstract. The concept of designated confirmer signatures was introduced by Chaum [Cha94] to improve a shortcoming of undeniable signatures. The present paper formalizes the definition of designated confirmer signatures and proves that a designated confirmer signature scheme is equivalent to a public-key encryption scheme with respect to existence. In addition, the paper proposes practical designated confirmer signature schemes which are more efficient in signing than the previous scheme [Cha94].

1 Introduction

The concept of *undeniable signatures* was proposed by Chaum et al. [Cha90, CA89]; the recipient of a signature cannot misuse the signature and the signer cannot subsequently deny the signature. Unfortunately, for many practical applications undeniable signatures have one major shortcoming compared to normal (self-authenticating) digital signatures. Since undeniable signatures rely on the signer cooperating in subsequent confirmations of the signature, if the signer should become unavailable, such as might be expected in the case of a default on the agreement authorized by the signature, or should refuse to cooperate, then the recipient cannot make use of the signature.

The concept of *designated confirmer signatures* was introduced by Chaum [Cha94] to solve this weakness of undeniable signatures. It involves three parties: the signer, recipient, and confirmer. In designated confirmer signature schemes, if the signer is unavailable to confirm the signature, the confirmer, previously designated by the signer, can confirm the signature for the recipient.

In [Cha94], however, no formal definition (i.e., no rigorous concept) of designated confirmer signatures was given, and only an example of designated confirmer signature based on the RSA scheme was proposed. As mentioned in [Cha94], the remaining problems were as follows:

- give a formal definition (i.e., rigorous concept) of designated confirmer signatures.
- construct a designated confirmer signature based on a more general assumption. (i.e., find a sufficient assumption which is as weak as possible.)
- clarify what assumption is essential for constructing a designated confirmer signature. (i.e., find a necessary assumption which is as strong as possible.)
- propose more efficient constructions than [Cha94].

Y.G. Desmedt (Ed.): Advances in Cryptology - CRYPTO '94, LNCS 839, pp. 61-74, 1994.

The present paper solves all these problems.

- This paper gives the first formal definition (i.e., rigorous concept) of designated confirmer signatures.
- This paper ultimately answers both the questions on necessary and sufficient assumptions. It proves that *designated confirmer signatures exist if and only if public-key encryption [GM84] exists.* That is, it shows that the existence of public-key encryption is the necessary and sufficient assumption for constructing designated confirmer signatures.
- This paper proposes practical designated confirmer signature schemes that are more efficient in signing than the previous scheme [Cha94]. The proposed schemes are based on three move identification protocols, while the previous scheme is based on the RSA scheme. Two typical constructions are shown, one of which utilizes the Schnorr scheme [Sch91] as a three move protocol, while the other uses instead the extended Fiat-Shamir scheme [GQ88, OhOk88] as a three move protocol.

The theoretical part of our results can be considered from the following viewpoint. In the theoretical research fields of cryptography, the relationships of (computational) cryptographic primitives have been investigated extensively for the latest ten years [1] and many typical cryptographic primitives have been classified into two classes: one-way function family (OWF) and encryption-decryption function family (EDF). OWF consists of the primitives that are equivalent to one-way functions with respect to existence. EDF consists of the primitives that seem to require the encryption-decryption property as well as the one-way property. Impagliazzo and Rudich's result [IR89] seems to imply that several primitives such as secret key agreement may essentially require the encryption-decryption property in addition to the one-way property. In other words, EDF seems to be exclusive to OWF. OWF includes digital signatures [NY89, Rom90], pseudo-random generators [ILL89, Has90], and bit-commitment [Nao90]. EDF includes secret key agreement, public-key encryption, and oblivious transfer. (Here note that the security of these primitives have been formally defined, and that, for example, a digital signature scheme means a digital signature scheme which is existentially secure against adaptive chosen message attacks [GMRi88].)

The following natural question is suggested: To which class, OWF or EDF, do undeniable signatures and designated confirmer signatures belong? Boyar et al [BCDP90] gave an answer to this question: undeniable signatures exist if and only if digital signatures exists (i.e., if and only if one-way functions exist [NY89, Rom90]). That is, undeniable signatures belong to OWF. Hence, the other remaining problem has been to determine to which class, OWF or EDF, designated confirmer signatures belong.

The present paper answers this problem. It shows that designated confirmer signatures belong to EDF. This result seems to be somewhat surprising since designated confirmer signatures belong to a class different from the class of undeniable signatures, although designated confirmer signatures were introduced as a variant of undeniable signatures. (On the contrary, the previous result that undeniable signatures belong to the same class as digital signatures belong to

[1] The relationships of physical (i.e., information theoretical, or unconditionally secure) cryptographic primitives have been also investigated (e.g., [Oka93]).

[BCDP90] is less surprising, since undeniable signatures were introduced as a variant of digital signatures.)

This paper is organized as follows. Section 2 formalizes designated confirmer signatures and proves the main theorem that implies the equivalence of the designated confirmer signatures and public-key encryption. Section 3 proposes practical designated confirmer signature schemes that are more efficient in signing than the previous scheme.

2 Theoretical Results

2.1 Definitions

In this section, we formally define the "designated confirmer signature", as a variant of the definition of the "undeniable signature" by [BCDP90]. In the definition of designated confirmer signatures, a "designated confirmer" is introduced in addition to the properties of the undeniable signatures.

Definition 1. [Secure designated confirmer signature scheme] A *secure* "designated confirmer signature" scheme is $(G_S, G_C, Sign, Conf_{(S,V)}, Conf_{(C,V)})$ such that the following conditions hold:

1. **Key generation (G_S, G_C):**
   Let S be a signer, and C be a designated confirmer. G_C is a probabilistic poly-time algorithm which, on input 1^n (the security parameter), outputs a pair of strings, (C's secret-key, C's public-key), which is denoted by $G_C(1^n) = (G1_C(1^n), G2_C(1^n))$.
   G_S is a probabilistic poly-time algorithm which, on input strings 1^n and C's public-key $\in G2_C(1^n)$, outputs a pair of strings, (S's secret-key, S's public-key), which is denoted by $G_S(1^n, G2_C(1^n)) = (G1_S(1^n, G2_C(1^n)), G2_S(1^n), G2_C(1^n))$. Hereafter, however, for simplicity, $G_S(1^n, G2_C(1^n)) = (G1_S(1^n, G2_C(1^n)), G2_S(1^n), G2_C(1^n))$ will be just written by $G_S(1^n) = (G1_S(1^n), G2_S(1^n))$. The probability is taken over G_C's and G_S's coin tosses. Note that, as a variant, the input of G_S can be restricted only to 1^n. (Then, a slight modification of the definitions of the signing and privacy is required in some cases, although it is fairly easy. See Note of this definition.)
2. **Signing ($Sign$):**
   $Sign$ is a probabilistic poly-time algorithm which, on input strings 1^n, m (message), C's public-key $\in G2_C(1^n)$, and S's (secret-key, public-key) $\in G_S(1^n)$, outputs a string ("designated confirmer signature"), which is denoted by $Sign(1^n, m, G_S(1^n))$ (shortly by $Sign(m)$). The probability is taken over $Sign$'s coin tosses. Let $\Sigma_S(m)$ be the set of $Sign(m)$.
3. **Confirmation and disavowal ($Conf_{(S,V)}, Conf_{(C,V)}$):**
   $Conf_{(S,V)}$ is an interactive proof [GMRa89] between S and V, which, on common input strings 1^n, m, s (the presumed signature of m), S's public-key $\in G2_S(1^n)$, and C's public-key $\in G2_C(1^n)$, outputs either 0 ("true") or 1 ("false"). Here, signer S is the prover with an auxiliary input, S's secret-key $\in G1_S(1^n)$, and V is the verifier. For all m, for any constant c, and for sufficiently large n,

$$\Pr(Conf_{(S,V)}(1^n, m, s, G2_S(1^n), G2_C(1^n)) = 0) > 1 - 1/n^c,$$

if $s = \Sigma_S(m)$, and

$$\Pr(Conf_{(S,V)}(1^n, m, s, G2_S(1^n), G2_C(1^n)) = 1) > 1 - 1/n^c,$$

otherwise. The probability is taken over the coin tosses of S and V.
$Conf_{(C,V)}$ is an interactive proof between C and V, which, on common input strings 1^n, m, s, S's public-key $\in G2_S(1^n)$, and C's public-key $\in G2_C(1^n)$, outputs either 0 ("true") or 1 ("false"). Here, designated confirmer C is the prover with an auxiliary input, C's secret-key $\in G1_C(1^n)$, and V is the verifier. For all m, for any constant c, and for sufficiently large n,

$$\Pr(Conf_{(C,V)}(1^n, m, s, G2_S(1^n), G2_C(1^n)) = 0) > 1 - 1/n^c,$$

if $s = \Sigma_S(m)$, and

$$\Pr(Conf_{(C,V)}(1^n, m, s, G2_S(1^n), G2_C(1^n)) = 1) > 1 - 1/n^c,$$

otherwise. The probability is taken over the coin tosses of C and V.

4. **Security:**
   - **[Security for signers: Unforgeability against any adaptively chosen message attacks]** Let F be a probabilistic poly-time forging algorithm which, on input strings 1^n, S's public-key $\in G2_S(1^n)$, C's public-key $\in G2_C(1^n)$, and C's secret-key $\in G1_C(1^n)$, can request and receive S's signatures of polynomially-many adaptively chosen messages $\{m_i\}$, can request the execution of $Conf_{(S,F)}$ for polynomially-many adaptively chosen strings (either true signature strings or fake signature strings), and finally outputs a pair of strings (m, s). Then, for all such F, for any constant c, for all sufficiently large n, the probability that F outputs (m, s) for which either $Conf_{(S,V)}$ or $Conf_{(C,V)}$ outputs 0 is less than $1/n^c$. The probability is taken over the coin tosses of G_S, G_C, $Sign$, S, V and F.
   - **[Security for confirmers]** Let A be a probabilistic poly-time attacking algorithm which, on input strings 1^n, S's public-key $\in G2_S(1^n)$, S's secret-key $\in G1_S(1^n)$, and C's public-key $\in G2_C(1^n)$, can request the execution of $Conf_{(C,A)}$ for polynomially-many strings (either true signature strings or fake signature strings) adaptively chosen by A, and finally execute $Conf_{(A,V)}$ which is accepted by V, where V is a honest verifier. Then, for all such A, for any constant c, for all sufficiently large n, the probability that A succeeds in the above-mentioned attack is less than $1/n^c$. The probability is taken over the coin tosses of G_S, G_C, C, V and A.

5. **Privacy (Untransferability):**
   - **[Signature simulator]** A *signature simulator*, relative to a verifier V, is a probabilistic polynomial time algorithm, which, when given a message m, outputs
   $Simulator(m) = (Fake(m), FakeT_{(S,V)}(Fake(m), m),$
   $FakeT_{(C,V)}(Fake(m), m))$,
   where $Fake(m)$ is a fake signature of m, $FakeT_{(S,V)}(Fake(m), m)$ is a simulated transcript of verifying conversation between the signer S and V proving that $Fake(m)$ is a valid signature, and $FakeT_{(C,V)}(Fake(m), m)$ is a simulated transcript of verifying conversation between the designated confirmer C and V proving that $Fake(m)$ is a valid signature.

- **[Signature oracle]** The *signature oracle* O, relative to a verifier V, receives a message m as input and outputs
  $Oracle(m) = (Sign(m), ValidT_{(S,V)}(Sign(m), m),$
  $ValidT_{(C,V)}(Sign(m), m))$,
  where $Sign(m)$ is a string chosen randomly from valid signatures of m, $ValidT_{(S,V)}(Sign(m), m)$ is a transcript chosen randomly from true verifying conversations between the true signer S and V proving that $Sign(m)$ is a valid signature, and $ValidT_{(C,V)}(Sign(m), m)$ is a transcript chosen randomly from true verifying conversations between the true designated confirmer C and V proving that $Sign(m)$ is a valid signature.
- **[Privacy (Untransferability)]** Let D be a polynomial time distinguisher, which is allowed to choose a message m, obtain some valid signatures of messages in set M', with $m \notin M'$, and interact with the true signer in verifying (and denying) the validity of the signatures of messages in set M'. Let $D(s, T_{(S,V)}, T_{(C,V)}, m)$ denote the output of a distinguisher D when its input is the possible signature s for message m, and possible transcripts $T_{(S,V)}$ and $T_{(C,V)}$. Let n be the security parameter.
  Then, for any verifier V, there exists a signature simulator relative to V such that, for any polynomial time distinguisher D, and for any constant c, the following holds for n sufficiently large:

$$|\Pr(D(Oracle(m), m) = 1) - \Pr(D(Simulator(m), m) = 1)| < 1/n^c.$$

Note: When the input of G_S is restricted only to 1^n, as a variant, a slight modification of the definitions of the signing and privacy is required in some schemes, as mentioned in the key generation. For example, $Sign(m)$ consists of two parts: one part ($Sign_1$) depends on C's public-key but not on m, and the other part ($Sign_2$) depends on m. Then, $Sign_1$ can be considered to be a part of S's public key, although it is not published before. (See the note in Step 1 of the signing protocol in the [If part] of the proof of Theorem 3.) Then, in the privacy definition, $Sign_2$ is considered to be $Sign(m)$ instead.

There are three kinds of definitions of a "secure" public-key encryption scheme [GM84], and these three definitions have been proven to be equivalent [MRS88]. Here, we adopt the definition based on the indistinguishability.

Definition 2. [Secure public-key encryption scheme] A *secure* "public-key encryption" scheme is (G, E, D) such that the following conditions hold:

1. **Key generation (G):**
   G is a probabilistic poly-time algorithm which, on input 1^n (the security parameter), outputs a pair of strings, (secret-key, public-key), which is denoted by $G(1^n) = (G1(1^n), G2(1^n))$. The probability is taken over G's coin tosses.
2. **Encryption (E):**
   E is a probabilistic poly-time algorithm which, on input strings 1^n, m (plaintext), public-key $\in G2(1^n)$, outputs a string ("ciphertext") c, which is denoted by $E_{G2(1^n)}(m)$. The probability is taken over E's coin tosses.
3. **Decryption (D):**

D is a probabilistic poly-time algorithm which, on input strings 1^n, ciphertext $c = E_{G2(1^n)}(m)$, (secret-key, public-key) $\in G_S(1^n)$, outputs a string, which is denoted by $D_{G(1^n)}(c)$. For any m, for any constant c, and for sufficiently large n,

$$\Pr(D_{G(1^n)}(E_{G2(1^n)}(m)) = m) > 1 - 1/n^c.$$

The probability is taken over D's coin tosses.

4. **Security:**
A public-key encryption scheme (G, E, D) is *secure* if for any polynomial sequence of random variables $X_n = (X_n^{(1)}, X_n^{(2)})$, for any polynomial time machine A, for any constant c and for any sufficiently large n

$$\Pr(X_n = (m_0, m_1)) \cdot (|\Pr(A((m_0, m_1), E_{G(1^n)}(m_0)) = 1)$$
$$- \Pr(A((m_0, m_1), E_{G(1^n)}(m_1)) = 1)|) < 1/n^c.$$

The probability is taken over $X_n^{(1)}$ and $X_n^{(2)}$'s distributions and coin tosses of A, G and E.

2.2 Equivalence of Designated Confirmer Signature and Public-Key Encryption

Theorem 3. [Main Theorem]
A secure designated confirmer signature scheme exists if and only if a secure public-key encryption scheme exists.

Sketch of Proof:

[If part:]

First, we assume the existence of a secure public-key encryption scheme.

Let $E_{e_C}(m)$ be a secure public-key encryption of a plaintext m for receiver C (e_C is C's public-key), where only C can decipher m by $m = D_{d_C}(E_{e_C}(m))$ (d_C is C's secret-key).

Then, an ordinary signature scheme which is existentially secure against adaptive chosen message attacks [GMRi88] (hereafter, we will call this signature scheme simply the "secure" signature scheme) exists, since a secure signature scheme exists if and only if a one-way function exists [NY89, Rom90], and a one-way function exists if a secure public-key encryption scheme exists (use the key generation function of the public-key encryption scheme to construct a one-way function).

Let $\sigma_S(m)$ be the set of ordinary secure signatures of m generated by signer S, and $V_S(m, s)$ be a verification boolean function for S's signature, where $V_S(m, s) = 0$ iff $s \in \sigma_S(m)$ and $V_S(m, s) = 1$ iff $s \notin \sigma_S(m)$.

We will now explain a designated confirmer protocol based on a secure public-key encryption scheme and ordinary secure signature scheme. Let S be a signer, C be a confirmer, and V be a verifier. The designated confirmer protocol consists of the signing protocol by S, confirmation protocol between S and V, confirmation protocol between C and V.

First the key generation and signing protocol between S and V is as follows:

Protocol: (Key generation and signing: designated confirmer signature)

Step 1 (Key generation) C generates a pair of keys, (e_C, d_C), and publishes e_C as C's public-key. S publishes a public function V_C as S's public-key for the verification of S's signature. S also publishes $P_{S,C} = E_{e_C}(K_C)$ for each designated confirmer C.
Note: Instead $P_{S,C}$ can be transmitted along with S's signature (i.e., certificate) in $\sigma_S(P_{S,C})$.

Step 2 (Signing) S generates a designated confirmer signature $Sign(m)$ of message m such that

$$Sign(m) = BC(s, h_{K_C}(m)),$$

where $s \in \sigma_S(m)$, BC is a secure bit commitment function [Nao90], and h is a pseudo-random function [GGM84]. S sends $(m, Sign(m))$ to V.

Note: A secure bit commitment function exists, if a secure public-key encryption scheme exists, since a secure bit commitment function exists when a one-way function exists [Nao90, ILL89, Has90]. A pseudo-random function exists, if a secure public-key encryption scheme exists. (This is also from the reduction to a one-way function [GGM84].)

The confirmation and disavowal protocol between S and V is as follows:

Protocol: (Confirmation and disavowal between S and V: designated confirmer signature)

Step 1 S determines, on input (m, Z), whether $Z \in \Sigma_S(m)$ or not, by the BC opening of Z with $h_{K_C}(m)$ (S can open the BC, since S knows K_C, i.e., $h_{K_C}(m)$).

Step 2 When S proves the validity of $Z = Sign(m) \in \Sigma_S(m)$, S proves to V that there exists (s, K_C, r) satisfying $V_S(m, s) = 0$, $Sign(m) = BC(s, h_{K_C}(m))$, and $P_{S,C} = E_{e_C}(K_C)$ with a zero-knowledge interactive proof for any NP problem [BCC88, IY87, GMW86], where r is a random string which is used for generating $E_{e_C}(K_C)$ from K_C. Note that such a zero-knowledge interactive proof exists since it is a poly-time predicate that (s, K_C, r) satisfies $V_S(m, s) = 0$, $Sign(m) = BC(s, h_{K_C}(m))$, and $P_{S,C} = E_{e_C}(K_C)$.

Step 3 When S proves that $Z \notin \Sigma_S(m)$ is an invalid signature of m, S proves to V either one of the followings with a zero-knowledge interactive proof [BCC88, IY87, GMW86]:

- there exists (K_C, r) such that $P_{S,C} = E_{e_C}(K_C)$ and the BC opening of Z with $h_{K_C}(m)$ is unsuccessful (i.e., $Z \neq BC(*, h_{K_C}(m))$).
- there exists (s', K_C, r) such that $P_{S,C} = E_{e_C}(K_C)$, $Z = BC(s', h_{K_C}(m))$ and $V_S(m, s') = 1$.

The confirmation and disavowal protocol between C and V is as follows:

Protocol: (Confirmation and disavowal between C and V: designated confirmer signature)

Step 1 C calculates $K_C = D_{d_C}(P_{S,C})$. C determines, on input (m, Z), whether $Z \in \Sigma_S(m)$ or not, by the BC opening of Z with $h_{K_C}(m)$.

Step 2 When C proves the validity of $Z = Sign(m) \in \Sigma_S(m)$, C proves to V that there exists (s, K_C, t) satisfying $V_S(m, s) = 0$, $Sign(m) = BC(s, h_{K_C}(m))$, and $G_t(1^n) = (e_C, d_C)$, $K_C = D_{d_C}(P_{S,C})$, with a zero-knowledge interactive proof [BCC88, IY87, GMW86], where t is a random string for key generation algorithm G to generate keys (e_C, d_C).

Step 3 When C proves that $Z \notin \Sigma_S(m)$ is an invalid signature of m, C proves to V either one of the followings with a zero-knowledge interactive proof [BCC88, IY87, GMW86]:

- there exists (K_C, t) such that $G_t(1^n) = (e_C, d_C)$, $K_C = D_{d_C}(P_{S,C})$, and the BC opening of Z with $h_{K_C}(m)$ is unsuccessful.
- there exists (s', K_C, t) such that $G_t(1^n) = (e_C, d_C)$, $K_C = D_{d_C}(P_{S,C})$, $Z = BC(s', h_{K_C}(m))$ and $V_S(m, s') = 1$.

Now, we show that the above-mentioned scheme satisfies the conditions for a secure designated confirmer signature scheme.

1. **Confirmation and disavowal:**
   - If $Z \in \Sigma_S(m)$, there exists (s, K_C, r) such that $V_S(m, s) = 0$, $Z = BC(s, h_{K_C}(m))$, and $P_{S,C} = E_{e_C}(K_C)$. Then, signer S knows (s, K_C, r) and prove verifier V that Z is S's valid signature of message m.
     If $Z \notin \Sigma_S(m)$, either one of the two cases described in the above protocol occurs. Then, S can prove that Z is not S's valid signature of message m.
   - Similarly, C, given (m, Z), can prove verifier V $Z \in \Sigma_S(m)$ and $Z \notin \Sigma_S(m)$ correctly.
2. **Security:**
   - If we assume that the proposed designated confirmer signature scheme does not satisfy the security condition for signers, then we can easily show that the underlying ordinary signature in $\sigma_S(m)$ is not *secure*. This is contradiction. Note that the zero-knowledge property of $Conf_{(S,V)}$ is used in this part. (Here, note that forger F can distinguish a true signature and false signature since F knows C's secret key.)
   - The security condition for confirmers is satisfied, from the zero-knowledge property of $Conf_{(C,V)}$. Note that attacker A can distinguish a true signature and false signature since A knows S's secret key.
3. **Privacy (Untransferability):**
   A signature simulator, *Simulator*, relative to V can be constructed as follows:
   - *Simulator* selects a random message a and calculates $E_{e_C}(a)$ as $Fake(m)$.
   - Let M_1 and M_2 be zero-knowledge simulators of confirmation protocols between S and V, and C and V, respectively. *Simulator* runs M_1 and sets $FakeT_{(S,V)}(Fake(m), m)$ = the output of M_1. *Simulator* also runs M_2 and sets $FakeT_{(C,V)}(Fake(m), m)$ = the output of M_2.

   From the definition of secure public-key encryption, $Fake(m)$ can be easily shown to be indistinguishable from $Sign(m)$. From the definition of zero-knowledge, $FakeT_{(S,V)}(Fake(m), m)$ and $FakeT_{(C,V)}(Fake(m), m)$ are indistinguishable from $ValidT_{(S,V)}(Sign(m), m)$, and $ValidT_{(C,V)}(Sign(m),$

m)), respectively. Following the well known Hybrid argument of [GM84], we conclude that $Oracle(m)$ and $Simulator(m)$ are indistinguishable.

[Only if part:]

We assume the existence of a secure designated confirmer signature scheme.

Then, the public-key encryption scheme can be constructed using the designated confirmer signature scheme as follows:

Protocol: (Public-key encryption)

Step 1 Key generation: The public-key of the designated confirmer in the underlying designated confirmer signature scheme is used for the public-key, e, of the encryption scheme. The corresponding secret-key of the confirmer is used for the secret-key, d, of the encryption scheme.

Step 2 Encryption: Suppose a plaintext, b, is one bit (0 or 1). An arbitrary message m is selected, and the other necessary parameters (signer's secret and public keys) for a signer in the underlying designated confirmer signature scheme is generated.
When $b = 0$, $E_e(b)$ is a valid designated confirmer signature, $Sign(m)$, of m along with the generated signer's parameters. When $b = 1$, $E_e(b)$ is a fake signature $Fake(m) \notin \Sigma_S(m)$, which is generated by the signature simulator $Simulator$, along with the signer's parameters. Here, note that $Fake(m) \notin \Sigma_S(m)$ can be checked by signer's confirmation protocol. Here, the public-key of the designated confirmer is used as e.

Step 3 Decryption: Execute the confirmation and disavowal protocol, $Conf_{(C,V)}$, for $E_e(b)$ as a designated confirmer signature of m. If the output of the protocol, $Conf_{(C,V)}$, is valid, $D_d(E_e(b))$ is 0. Otherwise, $D_d(E_e(b))$ is 1. Here, the secret-key of the designated confirmer is used as d.

Note: If the length of the plaintext is k bits, repeat the above procedure of encryption and decryption k times. (Note that the parameters of the signer can be shared.)

Now, we show that the above-mentioned scheme satisfies the conditions for a secure public-key encryption scheme.

First, from the property of the confirmation protocol of the designated confirmer, $D_d(E_e(b)) = b$ with overwhelming probability.

Next, we show that the above-mentioned public-key encryption scheme is *secure*. For simplicity of description, here we assume that a ciphertext is one bit. Then, $E_e(0)$ $(= Sign(m) \in \Sigma_S(m)$, m, public parameters) and $E_e(1)$ $(= Fake(m) \notin \Sigma_S(m)$, m, public parameters) are indistinguishable, since $Oracle(m)$ including $Sign(m)$ and $Simulator(m)$ including $Fake(m)$ are indistinguishable from the privacy condition of Definition 1.

□

Boyar et al [BCDP90] introduced the concept of convertible undeniable signatures as a variant of the undeniable signatures. The present paper shows that a convertible designated confirmer signature scheme can be constructed similarly, and that it is equivalent to the public-key encryption with respect to the existence.

Corollary 4. *A secure convertible designated confirmer signature scheme exists if and only if a secure public-key encryption scheme exists.*

Sketch of Proof:

[If part:] In addition to the protocols in Theorem 3 of the designated confirmer signature scheme, the *conversion* protocol [BCDP90] between C (or S) and V is as follows:

Protocol:

Step 1 C (or S) sends K_C to V.

Step 2 V calculates s by opening the bit commitment $Sign(m) = BC(s, h_{K_C}(m))$ through $h_{K_C}(m)$, and checks whether $V_S(m, s) = 0$ holds.

[Only if part:]

Same as this part of Theorem 3.

□

3 Practical Constructions

This section introduces a new type of practical constructions. The new schemes are more efficient in signing than Chaum's scheme [Cha94]. They are based on three move identification protocols such as Feige-Fiat-Shamir [FFS88], Schnorr [Sch91], the extended Fiat-Shamir [GQ88, OhOk88], and the modified Schnorr [Oka92], while Chaum's scheme is based on the RSA scheme. That is, many constructions are possible (e.g., FFS type, Schnorr type, etc.). Among these three move protocols, [Sch91] and [Oka92] are based on the discrete logarithm problem, while [FFS88] and [GQ88, OhOk88] are based on the factoring problem.

This section first gives general description which is common among these constructions is given. Next, two examples are given: one is based on the Schnorr scheme, and the other is based on the extended Fiat-Shamir scheme.

The advantage of the proposed schemes based on the discrete logarithm type protocols [Sch91, Oka92] compared to the Chaum scheme [Cha94] is:

- If the preprocessing technique is used in the signing stage, the time taken for signing is much shorter than with the Chaum scheme. That is, in the Chaum scheme, the running time for signing is at least as same as that for the RSA scheme (i.e., very slow), while, in the proposed scheme, the running time for signing after the preprocessing is negligible.
- The security of this construction depends on only one arithmetical problem, i.e., the discrete logarithm problem, while the Chaum scheme depends on two arithmetical problems, i.e., the discrete logarithm problem and factoring problem (if either one is breakable, the scheme is breakable, although our scheme is breakable only if the discrete logarithm problem is tractable). Apart from this security advantage, our scheme has some practical merits. One is that all required arithmetical procedures can be executed using the same modulus p and q. Another is that an elliptic curve variant can be constructed, which has practical merits such as shorter data size and less computational complexity.

- The [Oka92] variant of our scheme is provably secure (unforgeable against chosen message attacks) under fairly weak assumptions. On the contrary, the Chaum scheme depends on a very strong assumption that the RSA scheme with hash functions is unforgeable against chosen message attacks.

The advantage of the proposed schemes based on the factoring type protocols [FFS88, GQ88, OhOk88] over the Chaum scheme is:

- Even if the preprocessing technique is not used in the signing stage, signing is faster than with the Chaum scheme.
- Our schemes are provably secure (unforgeable against chosen message attacks) under fairly weak assumptions [FFS88, OhOk88, Oka92].

3.1 Basic Protocol

Let (A, D) be a three move identification protocol such that first S (prover) sends a message $x = A_1(w)$ to D (verfier), D sends e to S, S sends $y = A_2(w, e, s)$ to V, and finally D checks the validity of (x, e, y) by checking $x = D(e, y, a)$. Here w is random coin flips of S, a is S's public key, and s is S's secret key. H denotes a one-way hash function. (Theoretically, H should be an ideal random function, or a correlation-free one-way hash function [Oka92].)

We assume that signer S utilizes the functions (A_1, A_2, D) of a three move protocol (A, D), and S's public and secret keys are (a, s). Let $b = g^u \bmod p$ be confirmer C's public key and u be C's secret key. Here, p is a prime, and q is also a prime which divides $p - 1$. The order of g in the multiplicative group of Z_p^* is q.

First the signing protocol between S and V is as follows:

Protocol: (Signing and confirmation between S and V)

Step 1 S generates a designated confirmer signature (d, e, y) of message m such that

$$d = g^r \bmod p,\ e = (b^r \bmod p) \oplus H(m, x),\ x = A_1(w),\ y = A_2(w, e, s).$$

Here, $r \in_R Z_q$, and w is a random number. S sends $(m, (d, e, y))$ to V.

Step 2 S and V calculate

$$z = e \oplus H(m, D(e, y, a)).$$

S proves to V that $\log_g d = \log_b z$ in a zero-knowledge manner (without revealing r). Several efficient perfect zero-knowledge protocols have been known [Cha90, BCDP90].

The confirmation protocol between C and V is as follows:

Protocol: (Confirmation between C and V)

Step 1 C receives $(m, (d, e, y))$ from S or V. C and V calculate

$$z = e \oplus H(m, D(e, y, a)).$$

Step 2 C proves to V that $\log_g b = \log_d z$ in a zero-knowledge manner (without revealing u).

The conversion protocol between C and V is as follows:

Protocol: (Conversion by C)

Step 1 C calculates

$$z = e \oplus H(m, D(e, y, a)).$$

C proves to V that $\log_g b = \log_d z$ C sends (l_1, l_2, k) to V as follows:

$$l_1 = g^t \bmod p,\ l_2 = d^t \bmod p,\ k = t + H(l_1, l_2)u \bmod q.$$

Step 2 V checks whether

$$g^n \equiv l_1 b^{H(l_1,l_2)} \pmod p,\ d^n \equiv l_2 z^{H(l_1,l_2)} \pmod p$$

holds.

3.2 Example Based on Schnorr

Here, the Schnorr identification protocol [Sch91] is used as (A, D) such that first S (prover) sends a message $x = g^w \bmod p$ $(= A_1(w))$ to D (verfier), D sends $e \in Z_L$ to S, S sends $y = w + es \bmod q$ $(= A_2(w, e, s))$ to V, and finally D checks the validity of (x, e, y) by checking $x = g^y a^e \bmod p$ $(= D(e, y, a))$. Here $w \in_R Z_q$, $a = g^{-s} \bmod p$. (a: S's public key, s: S's secret key).

Let $b = g^u \bmod p$ be confirmer C's public key and u be C's secret key.

First the signing protocol between S and V is as follows:

Protocol: (Signing and confirmation between S and V)

Step 1 S generates a designated confirmer signature (d, e, y) of a message m such that

$$d = g^r \bmod p,\quad e = (b^r \bmod p) \oplus H(m, x),$$

$$x = g^w \bmod p,\quad y = w + es \bmod q.$$

S sends $(m, (d, e, y))$ to V.

Step 2 S and V calculate

$$z = e \oplus H(m, g^y a^e \bmod p).$$

S proves to V that $\log_g d = \log_b z$ in a zero-knowledge manner (without revealing r) by using [Cha90, BCDP90].

The confirmation protocol between C and V and the conversion protocol between C and V can be shown similarly.

3.3 Example Based on the Extended Fiat-Shamir

Here, the extended Fiat-Shamir identification protocol [GQ88, OhOk88] is used as (A, D) such that first S (prover) sends message $x = w^L \bmod n$ $(= A_1(w))$ to D (verfier), D sends $e \in Z_L$ to S, S sends $y = ws^e \bmod n$ $(= A_2(w, e, s))$ to V, and finally D checks the validity of (x, e, y) by checking $x = y^L a^e \bmod n$ $(= D(e, y, a))$. Here $w \in_R Z_n$, $a = 1/s^L \bmod n$, and $n = PQ$ (P, Q: primes). (a: S's public key, s: S's secret key).

Let $b = g^u \bmod p$ be confirmer C's public key and u be C's secret key.

First the signing protocol between S and V is as follows:

Protocol: (Signing and confirmation between S and V)

Step 1 S generates a designated confirmer signature (d, e, y) of a message m such that

$$d = g^r \bmod p, \quad e = (b^r \bmod p) \oplus H(m, x),$$

$$x = w^L \bmod n, \quad y = ws^e \bmod n.$$

Here, $r \in_R Z_q, w \in_R Z_n$. S sends $(m, (d, e, y))$ to V.

Step 2 S and V calculate

$$z = e \oplus H(m, y^L a^e \bmod n).$$

S proves to V that $\log_g d = \log_b z$ in a zero-knowledge manner (without revealing r) by using [Cha90, BCDP90].

The confirmation protocol between C and V and the conversion protocol between C and V can be shown similarly.

4 Conclusion

This paper has formalized the definition of designated confirmer signatures and has proven that a designated confirmer signature scheme is equivalent to a public-key encryption scheme with respect to existence. This paper also presented practical designated confirmer signature schemes which are more efficient in signing than the previous scheme.

Acknowledgments

The author wishes to thank anonymous referees for their useful suggestions, and is grateful to Kazuo Ohta for valuable discussions.

References

[BCC88] Brassard, G., Chaum, D., and Crépeau, C.: Minimum Disclosure Proofs of Knowledge. J. Computer and System Sciences, **37** (1988) 156–189

[BCDP90] Boyar, J., Chaum, D., Damgård, I., Pedersen, T.: Convertible Undeniable Signatures. Proc. of Crypto'90, LNCS **537**, Springer–Verlag, (1991) 189–205

[Cha90] Chaum, D.: Zero-Knowledge Undeniable Signatures. Proc. of Eurocrypto'90, LNCS **473**, Springer–Verlag, (1991) 458–464

[Cha94] Chaum, D.: Designated Confirmer Signatures. Proc. of Eurocrypt '94, LNCS, Springer–Verlag (to appear)

[CA89] Chaum, D., van Antwerpen, H.: Undeniable Signatures. Proc. of Crypto'89, LNCS **435**, Springer–Verlag, (1990) 212–216

[DH76] Diffie, W., Hellman, M. E.: New Directions in Cryptography. IEEE Trans. Information Theory, **22**, 6, (1976) 644–654

[ElG85] ElGamal, T.: A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. IEEE Trans. Information Theory, **31**, 4, (1985) 460–472

[FFS88] Feige, U., Fiat, A., Shamir, A.: Zero-Knowledge Proofs of Identity. J. of Cryptology, **1**, 2 (1988) 77–94

[GGM84] Goldreich, O., Goldwasser, S., Micali, S.: How to Construct Random Functions. J. of ACM, **33**, 4 (1984) 792–807

[GL89] Goldreich, O., Levin, L.: A Hard-Core Predicate for any One-way Function. Proc. of STOC'89 (1989) 25–32

[GM84] Goldwasser, S., Micali, S.: Probabilistic Encryption. J. Computer and System Sciences, **28**, 2 (1984) 270–299

[GMRa89] Goldwasser, S., Micali, S., Rackoff, C.: The Knowledge Complexity of Interactive Proof Systems. SIAM J. Comput., **18**, 1 (1989) 186–208

[GMRi88] Goldwasser, S., Micali, S., Rivest, R.: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. SIAM J. Comput., **17**, 2 (1988) 281–308

[GMW86] Goldreich, O., Micali, S., Wigderson, A.: Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design. Proc. FOCS (1986) 174–187

[GQ88] Guillou, L. C., Quisquater, J.J.: A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory. Proc. of Eurocrypt'88, LNCS **330**, Springer–Verlag (1988) 123–128

[Has90] Håstad, J.: Pseudo-Random Generators under Uniform Assumptions. Proc. of STOC (1990) 395–404

[ILL89] Impagliazzo, R., Levin, L., Luby, L.: Pseudo-Random Number Generation from One-Way Functions. Proc. of STOC (1989) 12–24

[IR89] Impagliazzo, R., Rudich, S.: Limits on the Provable Consequence of One-Way Permutations. Proc. of STOC (1989) 44–61

[IY87] Impagliazzo, R., Yung, M.: Direct Minimum-Knowledge Computations. Proc. of Crypto'87, LNCS **293**, Springer–Verlag (1988) 40–51

[MRS88] Micali, S., Rackoff, C., Sloan, B.: The Notion of Security of Probabilistic Cryptosystems. SIAM J. Comput., **17**, 2 (1988) 412–426

[Nao90] Naor, M.: Bit Commitment Using Pseudo-Randomness. Proc. of Crypto'89, LNCS **435**, Springer–Verlag, (1990) 128–136

[NY89] Naor, M., Yung, M.: Universal One-Way Hash Functions and Their Cryptographic Applications. Proc. of STOC (1989) 33–43

[OhOk88] Ohta, K., Okamoto, T.: A Modification of the Fiat-Shamir Scheme. Proc. of Crypto'88, LNCS **403**, Springer-Verlag (1990) 232-243

[Oka92] Okamoto, T.: Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. Proc. of Crypto'92, LNCS **740**, Springer–Verlag, (1993) 31–53

[Oka93] Okamoto, T.: On the Relationship among Cryptographic Physical Assumptions. Proc. of ISAAC'93, LNCS **762**, Springer–Verlag, (1993) 369–378

[Rom90] Rompel, J.: One-Way Functions are Necessary and Sufficient for Secure Signature. Proc. of STOC (1990) 387–394

[Sch91] Schnorr, C. P.: Efficient Signature Generation by Smart Cards. J. of Cryptology, **4**, 3 (1991) 161–174