ON THE SECURITY OF ADDING CONFIRMERS INTO DESIGNATED CONFIRMER SIGNATURES

Wataru Senga and Hiroshi Doi
Institute of Information Security, 2-14-1 Tsuruya-cho, Kanagawa-ku, Yokohama-shi, Kanagawa-ken, 221-0835, Japan

Keywords: Designated confirmer signature, Digital signature, Bilinear groups.

Abstract: In designated confirmer signature (DCS) scheme, a signature can be verified only by interacting with a semi-trusted third party, called the confirmer. In previous DCS schemes, a confirmer is designated at the time of the signature generation. So once the designated confirmer becomes unavailable, no one can verify the validity of the signature. In this paper, we introduce an extended DCS scheme which the confirmers can be added after the signature is generated. We give the new model and the security definitions, and propose the concrete scheme that is provably secure without random oracles.

1 INTRODUCTION

In ordinary digital signatures, anybody can verify the signature by using the public information. However, it might be undesirable that everyone can freely verify the signature according to the usage. To solve such a problem, Chaum and Antwerpen introduced the notion of *Undeniable Signatures* (Chaum and van Antwerpen, 1990). Undeniable signatures cannot be verified without the signer's cooperation unlike ordinary digital signatures. However, if the signer is not available, the verification of undeniable signatures becomes impossible. To overcome this problem, Chaum proposed the notion of *Designated Confirmer Signatures* (DCS) (Chaum, 1994), that is, a semi-trusted third party called the *confirmer* can convince the verifier that the signature is valid or invalid instead of a signer. In the DCS scheme, given a signature σ and a message m, the confirmer can prove that the signature is valid or invalid on a message m by executing a Confirm/Disavow protocol.

In previous DCS schemes, a confirmer is designated at the time of the signature generation. So once the designated confirmer becomes unavailable, no one can verify the validity of the signature. For example, consider the situation that the president of a company signs the document and wants to limit the verifiability. In such a case, his/her secretary can decide the appropriate verifier and convince a verifier of the validity of the signature using the DCS. However, if the secretary changes the section or retires from the company, anyone cannot convince a verifier that the signature is valid or invalid. From the above mentioned consideration, an extension of DCS scheme which the confirmers can be added dynamically is a preferable property.

In this paper we propose an extension of DCS scheme which the confirmers can be added after the signature is generated.

1.1 Related Work

After Chaum's proposal of the notion of DCS, many DCS schemes have been introduced (Okamoto, 1994; Michels and Stadler, 1998; Camenisch and Michels, 2000; Goldwasser and Waisbard, 2004; Gentry et al., 2005; Wang et al., 2007; Zhang et al., 2008).

Okamoto introduced a first formal model of DCS and showed that DCS and public key encryption are equivalent (Okamoto, 1994). Michels and Stadler pointed out that in the Okamoto's scheme the confirmer can forge valid signature on behalf of the signer. They also proposed new security model and constructed the concrete scheme (Michels and Stadler, 1998). Modifying the definitions of Okamoto (Okamoto, 1994) and Camenish and Michels (Camenisch and Michels, 2000), Goldwasser and Waisbard proposed a relaxed security definition and used strong witness hiding proof of knowledge instead of generic zero-knowledge proof in their Confirm protocol (Goldwasser and Waisbard, 2004). Gentry, Molnar and Ramzan presented a generic transformation to convert any secure signature scheme into DCS scheme (GMR scheme) without random oracles or

Senga W. and Doi H. (2009).
ON THE SECURITY OF ADDING CONFIRMERS INTO DESIGNATED CONFIRMER SIGNATURES.
In *Proceedings of the International Conference on Security and Cryptography*, pages 249-256
DOI: 10.5220/0002234702490256

generic zero-knowledge proofs (Gentry et al., 2005). Wang, Baek, Wong and Bao proposed the efficient DCS scheme improving the GMR scheme (Wang et al., 2007). Zhang, Chen and Wei proposed the simple and efficient DCS scheme based on bilinear pairing (Zhang et al., 2008).

1.2 Our Contribution

To construct a designated confirmer signature scheme which the confirmers can be added after the signature is generated (ADCS for short), introducing temporary public/secret confirmation keys (*pck*/*sck* for short) instead of confirmer's public/secret keys may be an effective strategy. According to the above strategy, anyone given the temporary *sck* can execute the Confirm/Disavow protocol. Although the required properties can be realized by introducing the *pck*/*sck* to the almost all existing DCS (Chaum, 1994; Okamoto, 1994; Michels and Stadler, 1998; Camenisch and Michels, 2000; Goldwasser and Waisbard, 2004; Gentry et al., 2005; Wang et al., 2007), a verifier cannot know who is the confirmer as long as auxiliary authentication protocols are not appended. Nevertheless, both Confirm/Disavow protocol based on these existing DCS are still complicated.

On the other hand, when we introduce the *pck*/*sck* to ZCW08 (Zhang et al., 2008) scheme, both Confirm/Disavow protocols become very simple. In ZCW08 scheme, it is easy to convert DCS into ordinary signature using *sck*. Furthermore it is easy to convert ordinary signature into DCS too. So the ADCS which should be verified using pck_1 can be easily modified to the ADCS which should be verified using pck_2. The illegal transformation may cause the serious trouble in some applications. So prohibiting such a transformation may be a preferable feature.

In this paper, we introduce a new model suitable for the above scenario and construct a new DCS scheme. Our scheme is an extension of ZCSM06 signature scheme (and similar to ZCW08 scheme) and has the following properties.

1. The Confirm/Disavow protocol is very simple.
2. It is impossible to transform the ADCS verifiable by pck_1 to the ADCS verifiable by pck_2.
3. The security of the proposed scheme in this model can be proved under the $k+1$-square roots assumption and l-BDHE assumption without random oracles.

2 PRELIMINARIES

We describe the settings and computational assumptions used in this paper.

2.1 Bilinear Groups

Let $\mathbb{G}$ and $\mathbb{G}_1$ be two (multiplicative) cyclic groups of prime order p and g be a generator of $\mathbb{G}$. A bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$ is said to be an admissible bilinear pairing if the following three conditions hold:

1. Bilinearity: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $e(g, g) \neq 1$, i.e. the map does not send all pairs in $\mathbb{G} \times \mathbb{G}$ to the identity in $\mathbb{G}_1$.
3. Computability: there is an efficient algorithm to compute $e(u, v)$ for all $u, v \in \mathbb{G}$.

2.2 $k+1$ Square Roots Assumption

The $k+1$ square roots problem in $(\mathbb{G}, \mathbb{G}_1)$ (Zhang et al., 2006) is stated as follows: given $\{k \in \mathbb{Z},\ x \in_R \mathbb{Z}_p,\ g \in \mathbb{G},\ \alpha = g^x,\ h_1, \ldots, h_k \in \mathbb{Z}_p, g^{(x+h_1)^{1/2}}, \ldots, g^{(x+h_k)^{1/2}}\}$ as input, output $(g^{(x+h)^{1/2}}, h \notin \{h_1, \ldots, h_k\})$.

Definition 1. *We say that the* $(k+1, t, \varepsilon)$*- square roots assumption holds in* $(\mathbb{G}, \mathbb{G}_1)$ *if no t-time algorithm has advantage at least* ε *in solving the* $k+1$ *square roots problem in* $(\mathbb{G}, \mathbb{G}_1)$.

2.3 l-BDHE Assumption

Let $\mathbb{G}$ be a bilinear group of prime order p. The l-BDHE problem (Boneh et al., 2005) in $\mathbb{G}$ is stated as follows: given $(h, g, g^{\alpha}, g^{\alpha^2}, g^{\alpha^l}, g^{\alpha^{l+2}}, \ldots, g^{\alpha^{2l}})$ as input, output $e(g, h)^{\alpha^{l+1}}$.

Definition 2. *We say that the* (t, ε, l)*-BDHE assumption holds in* $\mathbb{G}$ *if no* t*-time algorithm has advantage at least* ε *in solving the l-BDHE problem in* $\mathbb{G}$.

2.4 Zero Knowledge Proofs

In DCS schemes, a confirmer executes interactive protocol to prove a verifier that the signature is valid or invalid. We use a zero-knowledge proof of knowledge (ZKPoK) protocol rather than honest-verifier zero-knowledge protocol so that our scheme should be secure against arbitrary cheating verifier. In our scheme, only the simple ZKPoK (e.g. the equality or inequality of two discrete logarithms) are required. The special honest verifier ZKPoK of the equality (or inequality) of two discrete logarithms are well known (Camenisch and Shoup, 2003; Ogata et al., 2005), and the transformation techniques from such a special honest verifier ZKPoK into (concurrent) ZKPoK are well known too (Cramer et al., 2000; Damgård, 2000; Gennaro, 2004). Moreover, we need a knowledge extractor to prove the security in our scheme, and the fact that any ZKPoK protocol has a knowledge extractor is well known too. So we omit the description of the concrete ZKPoK protocols or the knowledge extractor in this paper.

3 MODEL AND DEFINITIONS

3.1 Outline of Our Model

The model of ADCS consists of a signer $\mathcal{S}$, confirmers (and candidates for confirmer) $\mathcal{C}_i(i = 1,2,\ldots,n)$ and a verifier $\mathcal{V}$.

In the previous DCS schemes, a signature is generated by using confirmer's public key and signer's secret (and public) key. However in ADCS model, a signer does not know who will become a confirmer in future, so we cannot use the ordinary DCS generation. In our construction, we introduce the confirmation key pair (the public confirmation key *pck* and the secret confirmation key *sck*). The secret confirmation key *sck* is necessary for the confirmer to confirm or disavow it.

To construct the scheme efficiently, the confirmer is regard as very highly trusted authority in our model. (Note that all designated confirmer signatures can be converted into ordinary ones by revealing *sck*.)

For the adversarial model, we classify the candidates for confirmer into two groups, namely, $\mathcal{SC}_h$*(selective honest confirmers)* and $\mathcal{SC}_c$*(selective corrupted confirmers)*. $\mathcal{C}_i \in \mathcal{SC}_h$ never reveal the secret key sk_i, and $\mathcal{C}_j \in \mathcal{SC}_c$ may reveal the secret key sk_j. This classification is similar to the selective ID secure IBE (Boneh and Boyen, 2004).

3.2 Formal Definitions

We describe the formal definitions of ADCS. Let $negl(\lambda)$ denote a negligible function; i.e., one that grows smaller than $1/\lambda^c$ for all c and all sufficiently large λ.

Definition 3. *A secure ADCS consists of following 8 algorithms:*

-KeyGen: *takes as input 1^λ and outputs some pairs of keys (sk_S, pk_S), (sk_i, pk_i), (sck, pck). sk_S is a signing key and pk_S is a verification key for $\mathcal{S}$. sk_i is a secret key and pk_i is a public key for $\mathcal{C}_i(i = 1,2,\ldots,n)$. sck is a secret confirmation key and pck is a public confirmation key.*

-Sign: *takes as input (m, sk_S) and outputs an ordinary signature $\tilde{\sigma}$ such that* $\mathsf{Verify}(m,\tilde{\sigma}, pk_S) =$Accept.

-Verify: *takes as input $(m,\tilde{\sigma}, pk_S)$ and output* Accept *if $\tilde{\sigma}$ is an output of* $\mathsf{Sign}(m, sk_S)$*, and output $\perp$ otherwise.*

-ConfirmedSign: *takes as input (m, sk_S, sck) and outputs an ADCS σ on m.*

-Extract: *releases the secret confirmation key sck. Once sck is released, all previous ADCSs become publicly verifiable.*

-Designate: *takes as input (sck, sk_i) and outputs pck_i, which is public confirmation key for $\mathcal{C}_i$. To be designated as a confirmer, $\mathcal{C}_i$ must receive the secret confirmation key from a signer or an existing confirmer who have already obtained a secret confirmation key.*

-Confirm: *is an interactive protocol between $\mathcal{C}_i$ and $\mathcal{V}$ with common input $(m,\sigma, pk_i, pck, pck_i)$. The output is $b \in \{\mathsf{Accept},\perp\}$. The protocol must be both complete and sound. For completeness, we require that there is some $\mathcal{C}_i$ such that if σ is a valid ADCS on m then $b = \mathsf{Accept}$. For soundness, we require that for all confirmers $\mathcal{C}'$ if σ is an invalid ADCS on m, then $\Pr[\mathsf{Confirm}(m,\sigma, pk_i, pck, pck_i) = \mathsf{Accept}] < negl(\lambda)$.*

-Disavow: *is an interactive protocol between $\mathcal{C}_i$ and $\mathcal{V}$ with common input $(m,\sigma, pk_i, pck, pck_i)$. The output is $b \in \{\mathsf{Accept},\perp\}$. The protocol must be both complete and sound. For completeness, we require that there is some $\mathcal{C}_i$ such that if σ is an invalid ADCS on m then $b = \mathsf{Accept}$. For soundness, we require that for all confirmers $\mathcal{C}'$ if σ is a valid ADCS on m, then $\Pr[\mathsf{Disavow}(m,\sigma, pk_i, pck, pck_i) = \mathsf{Accept}] < negl(\lambda)$.*

Actually the Extract algorithm should be rarely used because the influence is too large in our model. However, following the formal definitions of previous DCS (Camenisch and Michels, 2000; Goldwasser and Waisbard, 2004; Gentry et al., 2005; Wang et al., 2007; Zhang et al., 2008), we have left the Extract algorithm.

The primary condition of DCS is that nobody can confirm the validity of the signature except the confirmer. Furthermore, the security requirements are classified into two categories: *security for signers* and *security for confirmers*. Intuitively, *security for signers* guarantees that ADCSs are unforgeable under adaptive chosen message attacks and *security for confirmers* guarantees that no one except for confirmers can confirm the validity of ADCSs to verifiers.

We describe formal definitions of *security for signers* as follows:

Definition 4. (Security for signers) *An ADCS scheme is secure for signers if no probabilistic polynomial time adversary $\mathcal{A}$ has a non-negligible advantage in the following game:*

Game S

1. *The challenger $\mathcal{B}$ takes as input a security parameter 1^λ and gives $(pk_S, pk_1, ..., pk_n, pck)$ to $\mathcal{A}$.*
2. *The adversary $\mathcal{A}$ is permitted to a series of queries:*

   ConfirmedSign *queries: $\mathcal{A}$ submits a message m and receives an ADCS σ on m.*

   Extract *queries: $\mathcal{A}$ receives a secret confirmation key sck.*

   Designate *queries: $\mathcal{A}$ submits $\mathcal{C}_i$'s public key pk_i and receives a corresponding public confirmation key pck_i.*

   $\mathsf{Confirm}_{(\mathcal{A},\mathcal{V})}$/$\mathsf{Disavow}_{(\mathcal{A},\mathcal{V})}$ *queries: $\mathcal{A}$ executes* Confirm/Disavow *protocol in the confirmer role.*

   $\mathsf{Confirm}_{(\mathcal{C},\mathcal{A})}$/$\mathsf{Disavow}_{(\mathcal{C},\mathcal{A})}$ *queries: $\mathcal{A}$ executes* Confirm/Disavow *protocol in the verifier role.*

   $\mathsf{Corrupt}_{\mathcal{C}}$ *queries: $\mathcal{A}$ submits a confirmer's public key pk_i and receives a corresponding secret key sk_i.*
3. *At the end of this game, $\mathcal{A}$ outputs a pair (m^*, σ^*).*

$\mathcal{A}$ wins the game if $\mathsf{Confirm}_{(\mathcal{C},\mathcal{V})}$ $(m^*, \sigma^*, pk_i, pck, pck_i) = \mathsf{Accept}$ *such that m^* has never been queried to the* ConfirmedSign *oracle and that (m^*, σ^*) has never been accepted at the* $\mathsf{Confirm}_{(\mathcal{C},\mathcal{A})}$ *queries earlier. The $\mathcal{A}$'s advantage $\mathsf{Adv}^{\mathsf{S}}(\mathcal{A})$ is defined to be probability that $\mathcal{A}$ wins this game.*

Now we explain the *security for confirmers*. Informally, an ADCS scheme need to be secure against forgery and impersonation under adaptive chosen message attacks (Ogata et al., 2005). In other word, *security for confirmers* requires that no one, except legal confirmers, can generate a valid pair $(m, \sigma, pk_i^*, pck_i^*)$ which will be confirmed in Confirm protocol(here, m and σ may have already been produced by legal signer). In an ADCS scheme, the "legal" confirmers should know both the confirmer's secret key sk_i and the secret confirmation key sck. Therefore, *security for confirmers* is divided into the following two cases, i.e., 1)no one except for having sk_i can prove the validity of a DCS, and 2)no one except for having sck can prove the validity of a DCS.

We describe formal definitions of *security for confirmers* as follows:

Definition 5. (Security for confirmers) *An ADCS scheme is secure for confirmers if no probabilistic polynomial time adversary $\mathcal{A}$ has a non-negligible advantage in both of Game C-1 and Game C-2.*

Game C-1

1. *The adversary $\mathcal{A}$ classifies the candidates for confirmers $(\mathcal{C}_1, ..., \mathcal{C}_n)$ into $\mathcal{C}_i (i = 1, ..., l) \in \mathcal{SC}_h$ and $\mathcal{C}_j (j = l+1, ..., n) \in \mathcal{SC}_c$, and notifies the challenger $\mathcal{B}$.*
2. *$\mathcal{B}$ takes as input a security parameter 1^λ and gives $(pk_S, pk_1, ..., pk_n, pck)$ to $\mathcal{A}$.*
3. *$\mathcal{A}$ is permitted to a series of queries:*

   ConfirmedSign *queries: $\mathcal{A}$ submits a message m and receives an ADCS σ on m.*

   Extract *queries: $\mathcal{A}$ receives a secret confirmation key sck.*

   Designate *queries: $\mathcal{A}$ submits $\mathcal{C}_i \in \mathcal{SC}_h$'s public key pk_i and receives a corresponding public confirmation key pck_i for $\mathcal{C}_i$. Note that a query of $pk_j (\mathcal{C}_j \in \mathcal{SC}_c)$ is prohibited.*

   $\mathsf{Confirm}_{(\mathcal{A},\mathcal{V})}$/$\mathsf{Disavow}_{(\mathcal{A},\mathcal{V})}$ *queries: $\mathcal{A}$ executes* Confirm/Disavow *protocol in the confirmer role.*

   $\mathsf{Confirm}_{(\mathcal{C},\mathcal{A})}$/$\mathsf{Disavow}_{(\mathcal{C},\mathcal{A})}$ *queries: $\mathcal{A}$ executes* Confirm/Disavow *protocol in the verifier role with $\mathcal{C}_i \in \mathcal{SC}_h$. It is prohibited that $\mathcal{A}$ executes* Confirm/Disavow *protocol with $\mathcal{C}_j \in \mathcal{SC}_c$.*

   $\mathsf{Corrupt}_{\mathcal{S}}$ *queries: $\mathcal{A}$ submits a signer $\mathcal{S}$'s verification key pk_S and receives a corresponding signing key sk_S.*

   $\mathsf{Corrupt}_{\mathcal{C}}$ *queries: $\mathcal{A}$ submits a confirmer $\mathcal{C}_j \in \mathcal{SC}_c$'s public key pk_j and receives a corresponding secret key sk_S. Note that a query of $pk_i (\mathcal{C}_i \in \mathcal{SC}_h)$ is prohibited.*
4. *At the end of this game, $\mathcal{A}$ outputs a pair $(m, \sigma, pk_i^*, pck_i^*)$.*

$\mathcal{A}$ wins the game if $\mathsf{Confirm}_{(\mathcal{A},\mathcal{V})}$ $(m, \sigma, pk_i^*, pck, pck_i^*) = \mathsf{Accept}$. *The $\mathcal{A}$'s advantage $\mathsf{Adv}^{\mathsf{C-1}}(\mathcal{A})$ is defined to be probability that $\mathcal{A}$ wins this game.*

Game C-2

1. *The adversary $\mathcal{A}$ classifies the candidates for confirmers $(\mathcal{C}_1, ..., \mathcal{C}_n)$ into $\mathcal{C}_i (i = 1, ..., l) \in \mathcal{SC}_h$ and $\mathcal{C}_j (j = l+1, ..., n) \in \mathcal{SC}_c$, and notifies the challenger $\mathcal{B}$.*
2. *$\mathcal{B}$ takes as input a security parameter 1^λ and gives $(pk_S, pk_1, ..., pk_n, pck)$ to $\mathcal{A}$.*
3. *$\mathcal{A}$ is permitted to a series of queries:*

ConfirmedSign *queries: $\mathcal{A}$ submits a message m and receives an ADCS σ on m.*

Designate *queries: $\mathcal{A}$ submits $\mathcal{C}_i \in \mathcal{SC}_h$'s public key pk_i and receives a corresponding public confirmation key pck_i for $\mathcal{C}_i$. Note that a query of $pk_j(\mathcal{C}_j \in \mathcal{SC}_c)$ is prohibited.*

Confirm$_{(\mathcal{A},\mathcal{V})}$/Disavow$_{(\mathcal{A},\mathcal{V})}$ *queries: $\mathcal{A}$ executes* Confirm/Disavow *protocol in the confirmer role.*

Confirm$_{(\mathcal{C},\mathcal{A})}$/Disavow$_{(\mathcal{C},\mathcal{A})}$ *queries: $\mathcal{A}$ executes* Confirm/Disavow *protocol in the verifier role with $\mathcal{C}_i \in \mathcal{SC}_h$. It is prohibited that $\mathcal{A}$ executes* Confirm/Disavow *protocol with $\mathcal{C}_j \in \mathcal{SC}_c$.*

Corrupt$_{\mathcal{S}}$ *queries: $\mathcal{A}$ submits a signer $\mathcal{S}$'s verification key pk_S and receives a corresponding signing key sk_S.*

Corrupt$_{\mathcal{C}}$ *queries: $\mathcal{A}$ submits a confirmer $\mathcal{C}_j \in \mathcal{SC}_c$'s public key pk_j and receives a corresponding secret key sk_S. Note that a query of $pk_i(\mathcal{C}_i \in \mathcal{SC}_h)$ is prohibited.*

4. *At the end of this game, $\mathcal{A}$ outputs a pair $(m, \sigma, pk_i^*, pck_i^*)$.*

$\mathcal{A}$ wins the game if Confirm$_{(\mathcal{A},\mathcal{V})}$ $(m, \sigma, pk_i^*, pck, pck_i^*) =$ Accept. *The $\mathcal{A}$'s advantage* $\mathsf{Adv}^{\mathsf{C-2}}(\mathcal{A})$ *is defined to be probability that $\mathcal{A}$ wins this game.*

4 PROPOSED SCHEME

We present a construction of the scheme adding confirmer into DCS (ADCS) based on ZCSM06 signature scheme (Zhang et al., 2006).

4.1 The ZCSM06 Signature Scheme

We describe the ZCSM06 short signature scheme (Zhang et al., 2006).

Let $\mathbb{G}$ be bilinear groups where $|\mathbb{G}| = p$ for some prime p. Let g be a generator of $\mathbb{G}$.

KeyGen: pick random $x, y \in \mathbb{Z}_p^*$ and compute $u \leftarrow g^x$ and $v \leftarrow g^y$. The verification key is (u, v). The signing key is (x, y).

Sign: given a signing key (x, y) and a message $m \in \mathbb{Z}_p^*$, pick a random $r \in \mathbb{Z}_p^*$ and compute $\tilde{s} \leftarrow g^{(x+my+r)^{1/2}}$. Here $(x + my + r)^{1/2}$ is computed modulo p. In the unlikely event that $x + my + r$ is not a quadratic residue modulo p we try again with a different random r. The signature is $\tilde{\sigma} = (\tilde{s}, r)$.

Verify: given a verification key (u, v), a message m, and a signature $\tilde{\sigma} = (\tilde{s}, r)$, verify that

$$e(\tilde{s}, \tilde{s}) = e(uv^m g^r, g).$$

If the equality holds the result is valid; otherwise the result is invalid.

Theorem 1. *Suppose the $(k+1, t', \varepsilon')$-square roots assumption holds in $(\mathbb{G}, \mathbb{G}_1)$. Then the ZCSM06 signature scheme is (t, q_S, ε)-secure against existential forgery under a chosen message attack provided that*

$$q_S \leq k+1,\ \varepsilon = 2\varepsilon' + 4q_S/p \approx 2\varepsilon',\ t \leq t' - \Theta(q_S T).$$

4.2 Construction of ADCS Scheme

We describe a construction of ADCS scheme. The Sign and the Verify algorithm are same as ZCSM06, and the ConfirmedSign algorithm is similar to ZCW08 (Zhang et al., 2008).

Let $\mathbb{G}$ be a bilinear group where $|\mathbb{G}| = p$ for some prime p. Let g be a generator of $\mathbb{G}$.

KeyGen: pick random $x, y \in \mathbb{Z}_p^*$, random $k \in \mathbb{Z}_p^*$ such that k is a quadratic residue modulo p and compute $u \leftarrow g^x$, $v \leftarrow g^y$, $K \leftarrow g^k$ and $b \leftarrow g^{k^2}$. $(sk_S, pk_S) = ((x, y), (u, v))$, $(sck, pck) = (b, K)$. Pick random $a_i \in \mathbb{Z}_p^*$ $(i = 1, \ldots, n)$, and compute $A_i \leftarrow g^{a_i}$ $(i = 1, \ldots, n)$. $(sk_i, pk_i) = (a_i, A_i)$.

Sign: is same as the ZCSM06 signature scheme.

Verify: is same as the ZCSM06 signature scheme.

ConfirmedSign: given (x, y), k, and a message $m \in \mathbb{Z}_p^*$, pick a random $r \in \mathbb{Z}_p^*$ and compute $s \leftarrow K^{(x+my+r)^{1/2}}$. In the unlikely event that $x + my + r$ is not a quadratic residue modulo p we try again with a different random r. The ADCS is $\sigma = (s, r)$.

Extract: release $sck = b$. Once b is revealed, everyone can verify the signature σ on m by

$$e(s, s) = e(uv^m g^r, b).$$

If the equality holds the result is valid; otherwise the result is invalid.

Designate: given $sck = b$ and $sk_i = a_i$, computes $B_i \leftarrow b^{1/a_i}$, and discloses $pck_i = B_i$ as the public confirmation key for $\mathcal{C}_i$. Any verifier $\mathcal{V}$ can verify the validity of B_i by

$$e(K, K) = e(A_i, B_i).$$

Confirm: given a pair (m, σ), $\mathcal{C}_i$ verifies an ADCS by

$$e(s, s) = e(uv^m g^r, B_i)^{a_i}.$$

If the equality holds, $\mathcal{C}_i$ execute the interactive ZKPoK protocol with $\mathcal{V}$ as follows;

$$\mathrm{PK}\{(a_i) : e(s, s) = e(uv^m g^r, B_i)^{a_i} \wedge e(g, g)^{a_i} = e(A_i, g)\}.$$

otherwise, $\mathcal{C}_i$ outputs $\perp$.

Disavow: given a pair (m,σ), $\mathcal{C}_i$ verify an ADCS by

$$e(s,s) = e(uv^m g^r, B_i)^{a_i}.$$

If the equality does not hold, $\mathcal{C}_i$ execute the interactive ZKPoK protocol with $\mathcal{V}$ as follows;

$$\mathrm{PK}\{(a_i) : e(s,s) \neq e(uv^m g^r, B_i)^{a_i} \wedge e(g,g)^{a_i} = e(A_i, g)\}.$$

otherwise, $\mathcal{C}_i$ outputs $\perp$.

5 SECURITY

In this section, we prove security of the proposed scheme.

The security proof of the underlying scheme and our extension does not use random oracles. So, the proposed scheme can be proven without random oracles.

Theorem 2. *Suppose the ZCSM06 scheme is (t', q'_S, ε)-secure against existential forgery under a chosen message attack. Then the proposed scheme is (t,q,ε)-secure for signer provided that $t' = t + qT$ where q is the total number of queries that the adversary can issue to the oracles , T is a maximum time required to execute of an exponentiation in $\mathbb{G}$,* ConfirmedSign, Confirm$_{(\mathcal{A},\mathcal{V})}$, Disavow$_{(\mathcal{A},\mathcal{V})}$, Confirm$_{(\mathcal{C},\mathcal{A})}$, Disavow$_{(\mathcal{C},\mathcal{A})}$ *queries.*

Proof. Let $\mathcal{A}$ be a PPT adversary that has non-negligible advantage $\mathsf{Adv}^{\mathsf{S}}(\mathcal{A})$. We construct a simulator $\mathcal{B}$ which forges the ZCSM06 signature using $\mathcal{A}$.
Let $(\mathbb{G}, \mathbb{G}_1, e, p, g)$ be a parameter of bilinear groups. $\mathcal{B}$ is given a pair $(g, u = g^x, v = g^y)$ generated by ZCSM06's KeyGen algorithm. $\mathcal{B}$ picks a random $k \in \mathbb{Z}_p^*$ and compute $K \leftarrow g^k, b \leftarrow g^{k^2}$. $\mathcal{B}$ also picks a random $a_i \in \mathbb{Z}_p^* (i = 1,\ldots,n)$ and compute $A_i \leftarrow g^{a_i} (i = 1,\ldots n)$. $\mathcal{A}$ is given $(g,u,v,K,A_i)(i = 1,\ldots,n)$. $\mathcal{A}$ makes queries adaptively, and $\mathcal{B}$ responds as follows:

When $\mathcal{A}$ makes a ConfirmedSign query for a message m, $\mathcal{B}$ queries ZCSM06's signing oracle with the same m. Then $\mathcal{B}$ obtains $\tilde{\sigma} = (\tilde{s}, r)$, computes $s \leftarrow \tilde{s}^k$, and returns $\sigma = (s,r)$ to $\mathcal{A}$.

When $\mathcal{A}$ makes a Confirm$_{(\mathcal{A},\mathcal{V})}$ / Disavow$_{(\mathcal{A},\mathcal{V})}$ query, $\mathcal{B}$ performs the Confirm/Disavow protocol in the verifier role. $\mathcal{B}$ need not know any secret information.

When $\mathcal{A}$ makes a Confirm$_{(\mathcal{C},\mathcal{A})}$ / Disavow$_{(\mathcal{C},\mathcal{A})}$ query, $\mathcal{B}$ performs the Confirm/Disavow protocol in the confirmer(prover) role. $\mathcal{B}$ can perform the protocol because $\mathcal{B}$ has all secret information for the confirmer.

When $\mathcal{A}$ makes a Designate query for a confirmer $\mathcal{C}_i$, $\mathcal{B}$ returns $B_i = b^{1/a_i}$ to $\mathcal{A}$.

When $\mathcal{A}$ makes a Extract query, $\mathcal{B}$ returns the secret confirmation key b to $\mathcal{A}$.

When $\mathcal{A}$ makes a Corrupt$_C$ query for a confirmer $\mathcal{C}_i$, $\mathcal{B}$ returns a_i to $\mathcal{A}$.

$\mathcal{B}$ does not abort during above simulation, and finally $\mathcal{A}$ outputs (m^*, s^*, r^*) such that $e(s^*, s^*) = e(uv^{m^*} g^{r^*}, b)$. Let $\tilde{s}^* \leftarrow s^{*1/k}$, and $\mathcal{B}$ outputs $(m^*, \tilde{s}^*, r^*)$. Because of $e(\tilde{s}^*, \tilde{s}^*) = e(uv^{m^*} g^{r^*}, g)$, $\mathcal{B}$ succeeds in forgery on ZCSM06 signature scheme . $\square$

Theorem 3. *Suppose the $(t', \varepsilon, 3)$-BDHE assumption holds in $\mathbb{G}$. Then the proposed scheme is (t,q,ε)-secure for confirmer.*

To provide the proof of Theorem 3, we show Lemma 1 and Lemma 2.

Lemma 1. *If there exists a t-time algorithm $\mathcal{A}$ which satisfies $\mathsf{Adv}^{\mathsf{C-1}}(\mathcal{A}) \geq \varepsilon$, there exists an algorithm which solves $(t', \varepsilon, 3)$-BDHE problem. Here $t' = t + qT$ where q is the total number of queries that the adversary can issue to the oracles and T is a maximum time required to execute of an exponentiation in $\mathbb{G}$,* ConfirmedSign, Confirm$_{(\mathcal{A},\mathcal{V})}$, Disavow$_{(\mathcal{A},\mathcal{V})}$, Confirm$_{(\mathcal{C},\mathcal{A})}$, Disavow$_{(\mathcal{C},\mathcal{A})}$ *queries.*

Proof. Let $\mathcal{A}$ be a PPT adversary that has non-negligible advantage $\mathsf{Adv}^{\mathsf{C-1}}(\mathcal{A})$. We construct a simulator $\mathcal{B}$ which solves 3-BDHE problem using $\mathcal{B}$.

$\mathcal{B}$ is given a random 3-BDHE challenge $(H, G, G^z, G^{z^2}, G^{z^3}, G^{z^5}, G^{z^6})$.

$\mathcal{A}$ outputs the list of $\mathcal{C}_i \in \mathcal{SC}_h$ (the identity of selective honest confirmers) and $\mathcal{C}_j \in \mathcal{SC}_c$ (the identity of selective corrupted confirmers), and notifies $\mathcal{B}$.

Let $g \leftarrow G^{z^2}$ and $h \leftarrow G^{z^3}$. $\mathcal{B}$ picks random values $(x,y) \in \mathbb{Z}_p^*$ as a signing key and computes $(u \leftarrow g^z, v \leftarrow g^y)$ as a verification key. $\mathcal{B}$ picks a random value $k \in \mathbb{Z}_p^*$ and computes $b \leftarrow g^{k^2}$ as a secret confirmation key and $K \leftarrow g^k$ as a public confirmation key. Furthermore, $\mathcal{B}$ generates confirmer's public and secret key pairs as follows:

- For $\mathcal{C}_i \in \mathcal{SC}_h$, $\mathcal{B}$ picks random values $a_i (i = 1,\ldots,l)$ as secret keys and computes $A_i \leftarrow (g^z)^{a_i} (i = 1,\ldots,l)$ as public keys.
- For $\mathcal{C}_j \in \mathcal{SC}_c$, $\mathcal{B}$ picks random values $a_j (j = l+1,\ldots,n)$ as secret keys and computes $A_j \leftarrow g^{a_j} (j = l+1,\ldots,n)$ as public keys.

$\mathcal{A}$ is given the (u,v), $(A_1,\ldots,A_l)$, $(A_{l+1},\ldots,A_n)$, K. $\mathcal{A}$ makes queries adaptively, and $\mathcal{B}$ responds as follows:

When $\mathcal{A}$ makes a ConfirmedSign query for a message m, $\mathcal{B}$ picks a random $r \in \mathbb{Z}_p^*$ and computes $s \leftarrow K^{(x+my+r)^{1/2}}$. Then $\mathcal{B}$ returns $\sigma = (s,r)$ to $\mathcal{A}$.

When $\mathcal{A}$ makes a $\mathsf{Confirm}_{(\mathcal{A},\mathcal{V})}$ / $\mathsf{Disavow}_{(\mathcal{A},\mathcal{V})}$ query, $\mathcal{B}$ performs the Confirm/Disavow protocol in the verifier role. $\mathcal{B}$ need not know any secret information.

When $\mathcal{A}$ makes a $\mathsf{Confirm}_{(\mathcal{C},\mathcal{A})}$ / $\mathsf{Disavow}_{(\mathcal{C},\mathcal{A})}$ query, $\mathcal{B}$ performs the Confirm/Disavow protocol in the confirmer(prover) role. Note that $\mathcal{B}$ can perform in the verifier role by rewinding since the protocol is ZKPoK.

When $\mathcal{A}$ makes a Designate query for a confirmer $\mathcal{C}_i$, $\mathcal{B}$ computes $B_i \leftarrow (G^z)^{k^2/a_i} = b^{1/(za_i)}$, and returns B_i to $\mathcal{A}$.

When $\mathcal{A}$ makes a Extract query, $\mathcal{B}$ returns the secret confirmation key $b(=g^{k^2})$ to $\mathcal{A}$.

When $\mathcal{A}$ makes a $\mathsf{Corrupt}_{\mathcal{S}}$ query, $\mathcal{B}$ returns x,y to $\mathcal{A}$.

When $\mathcal{A}$ makes a $\mathsf{Corrupt}_{\mathcal{C}}$ query for $\mathcal{C}_j \in \mathcal{SC}_c$, $\mathcal{B}$ returns a_j to $\mathcal{A}$. The query for a confirmer $\mathcal{C}_i \in \mathcal{SC}_h$ is prohibited.

$\mathcal{B}$ does not abort during above simulation, and finally output a pair $(m^*, s^*, r^*, A_{i^*}, B_{i^*})$ which is accepted in Confirm protocol. Here, simulator $\mathcal{B}$ can obtain $\log_g A_{i^*} = za_{i^*}$ by using the knowledge extractor and can get z(because $\mathcal{B}$ generated a_{i^*}). $\mathcal{B}$ computes $G^{z^4} \leftarrow (G^{z^3})^z$, then outputs $e(G^{z^4}, H) = e(G,H)^{z^4}$, that is, $\mathcal{B}$ solves 3-BDHE problem. $\square$

Lemma 2. *If there exists a t-time algorithm $\mathcal{A}$ which satisfies $\mathsf{Adv}^{\mathsf{C-2}}(\mathcal{A}) \geq \varepsilon$, there exists an algorithm which solves $(t', \varepsilon, 3)$-BDHE problem. Here $t' = t + qT$ where q is the total number of queries that the adversary can issue to the oracles and T is a maximum time required to execute of an exponentiation in $\mathbb{G}$ or ConfirmedSign or $\mathsf{Confirm}_{(\mathcal{A},\mathcal{V})}$/$\mathsf{Disavow}_{(\mathcal{A},\mathcal{V})}$ or $\mathsf{Confirm}_{(\mathcal{C},\mathcal{A})}$/$\mathsf{Disavow}_{(\mathcal{C},\mathcal{A})}$ queries.*

Proof. Let $\mathcal{A}$ be a PPT adversary that has non-negligible advantage $\mathsf{Adv}^{\mathsf{game2}}(\mathcal{A})$. We construct a simulator $\mathcal{B}$ which solves 3-BDHE problem using $\mathcal{B}$.

$\mathcal{B}$ is given a random 3-BDHE challenge $(h, g, g^z, g^{z^2}, g^{z^3}, g^{z^5}, g^{z^6})$.

$\mathcal{A}$ outputs the list of $\mathcal{C}_i \in \mathcal{SC}_h$ (the identity of selective honest confirmers) and $\mathcal{C}_j \in \mathcal{SC}_c$ (the identity of selective corrupted confirmers), and notifies $\mathcal{B}$.

$\mathcal{B}$ picks a random pair $(x,y) \in \mathbb{Z}_p^*$ as a signing key and compute $(u \leftarrow g^x, v \leftarrow g^y)$ as a verification key. $\mathcal{B}$ sets $K \leftarrow g^{z^2}$ as a public confirmation key. Furthermore, $\mathcal{B}$ generates confirmer's public and secret key pairs as follows:

- For $\mathcal{C}_i \in \mathcal{SC}_h$, $\mathcal{B}$ picks random values $a_i (i = 1, \ldots, l)$ as secret keys and computes $A_i \leftarrow (g^z)^{a_i} (i = 1, \ldots, l)$ as public keys.
- For $\mathcal{C}_j \in \mathcal{SC}_c$, $\mathcal{B}$ picks random values $a_j (j = l+1, \ldots, n)$ as secret keys and computes $A_j \leftarrow g^{a_j} (j = l+1, \ldots, n)$ as public keys.

$\mathcal{A}$ is given the (u,v), $(A_1, \ldots, A_l)$, $(A_{l+1}, \ldots, A_n)$, K. $\mathcal{A}$ makes queries adaptively, and $\mathcal{B}$ responds as follows:

When $\mathcal{A}$ makes a ConfirmedSign query for a message m, $\mathcal{B}$ picks a random $r \in \mathbb{Z}_p^*$ and computes $s \leftarrow K^{(x+my+r)^{1/2}}$. Then $\mathcal{B}$ returns $\sigma = (s,r)$ to $\mathcal{A}$.

When $\mathcal{A}$ makes a $\mathsf{Confirm}_{(\mathcal{A},\mathcal{V})}$ / $\mathsf{Disavow}_{(\mathcal{A},\mathcal{V})}$ query, $\mathcal{B}$ performs the Confirm/Disavow protocol in the verifier role. $\mathcal{B}$ need not know any secret information.

When $\mathcal{A}$ makes a $\mathsf{Confirm}_{(\mathcal{C},\mathcal{A})}$ / $\mathsf{Disavow}_{(\mathcal{C},\mathcal{A})}$ query, $\mathcal{B}$ performs the Confirm/Disavow protocol in the confirmer(prover) role. Note that $\mathcal{B}$ can perform in the verifier role by rewinding since the protocol is ZKPoK. The query that $\mathcal{A}$ interacts with $\mathcal{C}_j \in \mathcal{SC}_c$ is prohibited.

When $\mathcal{A}$ makes a Designate query for a confirmer $\mathcal{C}_i \in \mathcal{SC}_h$, $\mathcal{B}$ computes $B_i \leftarrow (g^{z^3})^{1/a_i}$, and returns B_i to $\mathcal{A}$. The query for a confirmer $\mathcal{C}_i \in \mathcal{SC}_c$ is prohibited.

When $\mathcal{A}$ makes a $\mathsf{Corrupt}_{\mathcal{S}}$ query, $\mathcal{B}$ returns x,y to $\mathcal{A}$.

When $\mathcal{A}$ makes a $\mathsf{Corrupt}_{\mathcal{C}}$ query for $\mathcal{C}_j \in \mathcal{SC}_c$, $\mathcal{B}$ returns a_j to $\mathcal{A}$. The query for a confirmer $\mathcal{C}_i \in \mathcal{SC}_h$ is prohibited.

$\mathcal{B}$ does not abort during above simulation. Finally, $\mathcal{A}$ output a pair $(m^*, s^*, r^*, A_{i^*}, B_{i^*})$ which is accepted in $\mathsf{Confirm}_{(\mathcal{A},\mathcal{V})}$. Note that the equation $e(K,K) = e(A_{i^*}, B_{i^*})$ holds.

Here, simulator $\mathcal{B}$ can obtain $\log_g A_{i^*}$ using the knowledge extractor. Let α_i be $\log_g A_{i^*}$. $e(K,K) = e(g, g^{z^4})$ holds. On the other hand, $e(A_{i^*}, B_{i^*}) = e(g^{\alpha_i}, B_{i^*}) = e(g, B_{i^*}^{\alpha_i})$ holds.

Hence, $\mathcal{B}$ gets $B_{i^*}^{\alpha_i} = g^{z^4}$ and computes $e(g^{z^4}, h) = e(g,h)^{z^4}$. So, $\mathcal{B}$ solves 3-BDHE problem. $\square$

6 CONCLUSIONS

In this paper, we have shown new designated confirmer signature scheme, named ADCS, which the confirmers can be added after the signature is generated. For this purpose we gave the new model and the security definitions. Our concrete scheme shown in Section.4 accomplishes the *security for signers* and the *security for confirmers* in the standard model.

Note that our model has some restrictions (e.g. $\mathcal{SC}_h$ and $\mathcal{SC}_c$ should be decided before the adversarial game). It may be an interesting work to remove the restrictions.

In our scenario, the confirmers have broad powers which can freely transfer their ability of signature verification. For some practical purposes, the following improvement might be an interesting work too.

- The person who can designate new confirmers is not the confirmer but the original signer (or an alternative authority).
- The number of times of Designate by the confirmers is limited.

The above extensions are open problems.

ACKNOWLEDGEMENTS

We are grateful to Ryoichi Sasaki and Taiichi Saito for helpful discussions and suggestions on this work. We would also like to thank the anonymous reviewers for their insightful comments.

REFERENCES

Boneh, D. and Boyen, X. (2004). Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 223–238. Springer-Verlag.

Boneh, D., Boyen, X., and Goh, E.-J. (2005). Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer-Verlag.

Camenisch, J. and Michels, M. (2000). Confirmer signature schemes secure against adaptive adversaries. In *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 243–258. Springer-Verlag.

Camenisch, J. and Shoup, V. (2003). Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO 2003*, volume 2729 of *LNCS*, pages 126–144. Springer-Verlag.

Chaum, D. (1994). Designated confirmer signatures. In *EUROCRYPT 1994*, volume 950 of *LNCS*, pages 86–91. Springer-Verlag.

Chaum, D. and van Antwerpen, H. (1990). Undeniable signatures. In *CRYPTO 1989*, volume 435 of *LNCS*, pages 212–216. Springer-Verlag.

Cramer, R., Damgård, I., and MacKenzie, P. (2000). Efficient zero-knowledge proofs of knowledge without intractability assumptions. In *PKC 2000*, volume 1751 of *LNCS*, pages 354–373. Springer-Verlag.

Damgård, I. (2000). Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 418–430. Springer-Verlag.

Gennaro, R. (2004). Multi-trapdoor commitments and their applications to proofs of knowledge secure under concurrent man-in-the-middle attacks. In *CRYPTO 2004*, volume 3152 of *LNCS*, pages 220–236. Springer-Verlag.

Gentry, C., Molnar, D., and Ramzan, Z. (2005). Efficient designated confirmer signatures without random oracles or general zero-knowledge proofs. In *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 662–681. Springer-Verlag.

Goldwasser, S. and Waisbard, E. (2004). Transformation of digital signature schemes into designated confirmer signature schemes. In *TCC 2004*, volume 2951 of *LNCS*, pages 77–100. Springer-Verlag.

Michels, M. and Stadler, M. (1998). Generic constructions for secure and efficient confirmer signature schemes. In *EUROCRYPT 1998*, volume 1403 of *LNCS*, pages 406–421. Springer-Verlag.

Ogata, W., Kurosawa, K., and Heng, S.-H. (2005). The security of the fdh variant of chaum's undeniable signature scheme. In *PKC 2005*, volume 3386 of *LNCS*, pages 328–345. Springer-Verlag.

Okamoto, T. (1994). Designated confirmer signatures and public-key encryption are equivalent. In *CRYPTO 1994*, volume 839 of *LNCS*, pages 61–74. Springer-Verlag.

Wang, G., Baek, J., Wong, D. S., and Bao, F. (2007). On the generic and efficient constructions of secure designated confirmer signatures. In *PKC 2007*, volume 4450 of *LNCS*, pages 43–60. Springer-Verlag.

Zhang, F., Chen, X., Susilo, W., and Mu, Y. (2006). A new signature scheme without random oracles from bilinear pairings. In *VIETCRYPT 2006*, volume 4341 of *LNCS*, pages 67–80. Springer-Verlag.

Zhang, F., Chen, X., and Wei, B. (2008). Efficient designated confirmer signature from bilinear pairings. In *ASIACCS '08: Proceedings of the 2008 ACM symposium on Information, computer and communications security*, pages 363–368. ACM.