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 \( \sigma \) 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, Molver and Ramzan presented a generic transformation to convert any secure signature scheme into DCS scheme (GMR scheme) without random oracles or...
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 confirmer can be added after the signature is generated (ADCS for short), introducing temporary public/private confirmation keys (pck/sck for short) instead of confirmer’s public/private 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 pck1 can be easily modified to the ADCS which should be verified using pck2. 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 pck1 to the ADCS verifiable by pck2.
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 $G$ and $G_1$ be two (multiplicative) cyclic groups of prime order $p$ and $g$ be a generator of $G$. A bilinear map $e : G \times G \rightarrow G_1$ is said to be an admissible bilinear pairing if the following three conditions hold:

1. Bilinearity: for all $u, v \in G$ 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 $G \times G$ to the identity in $G_1$.
3. Computability: there is an efficient algorithm to compute $e(u, v)$ for all $u, v \in G$.

2.2 $k + 1$ Square Roots Assumption

The $k + 1$ square roots problem in $(G, G_1)$ (Zhang et al., 2006) is stated as follows: given $\{k \in \mathbb{Z}, x \in \mathbb{Z}_p, g \in G, \alpha = g^x, h_1, ..., h_k \in \mathbb{Z}_p, g^{(x+h_1)^2}, ..., g^{(x+h_k)^2}\}$ as input, output $(g^{(x+h)^2}, h \notin \{h_1, ..., h_k\})$.

Definition 1. We say that the $(k + 1, t, \varepsilon)$-square roots assumption holds in $(G, G_1)$ if no $t$-time algorithm has advantage at least $\varepsilon$ in solving the $k + 1$ square roots problem in $(G, G_1)$.

2.3 $l$-BDHE Assumption

Let $G$ be a bilinear group of prime order $p$. The $l$-BDHE problem (Boneh et al., 2005) in $G$ is stated as follows: given $(h, g, g_1^a, g_2^{a'}, g_1^{a''}, g_2^{a'''}, ..., g_1^{a^k})$ as input, output $e(g, h)^{a^l}$.

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

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 $s$, confirmer (and candidates for confirmer) $c_i (i = 1, 2, ..., n)$ and a verifier $\nu'$. 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 regarded 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, $S_{ch}$ (selective honest confirmer) and $S_{c}$ (selective corrupted confirmer). $c_i \in S_{ch}$ never reveal the secret key $sk_i$, and $c_i \in S_{c}$ may reveal the secret key $sk_i$. 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 $\text{negl}(\lambda)$ denote a negligible function; i.e., one that grows smaller than $1/\lambda^2$ for all $\lambda$ and all sufficiently large $\lambda$.

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

- **KeyGen**: takes as input $1^\lambda$ and outputs some pairs of keys $(sk_s, pk_s)$, $(sk_i, pk_i)$, $(sk_c, pck)$. $sk_s$ is a signing key and $pk_s$ is a verification key for $s$. $sk_i$ is a secret key and $pk_i$ is a public key for $c_i (i = 1, 2, ..., n)$. $sck$ is a secret confirmation key and $pck$ is a public confirmation key.
- **Sign**: takes as input $(m, sk_s)$ and outputs an ordinary signature $\sigma$ such that $\text{Verify}(m, \sigma, pk_s) = \text{Accept}$.
- **Verify**: takes as input $(m, \sigma, pk_s)$ and output $\text{Accept}$ if $\sigma$ is an output of $\text{Sign}(m, sk_s)$, and output $\bot$ otherwise.
- **ConfirmedSign**: takes as input $(m, sk_s, sck)$ and outputs an ADCS $\sigma$ 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 $c_i$.
- **To be designated as a confirmer, $c_i$ 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 $c_i$ and $\nu'$ with common input $(m, \sigma, pk_i, pck, pck_i)$. The output is $b \in \{\text{Accept}, \bot\}$. The protocol must be both complete and sound. For completeness, we require that there is some $c_i$ such that if $\sigma$ is a valid ADCS on $m$ then $b = \text{Accept}$. For soundness, we require that for all $c_i$ of $\sigma$ is an invalid ADCS on $m$, then $\text{Prob}[(\text{Confirm}(m, \sigma, pk_i, pck, pck_i) = \text{Accept}) < \text{negl}(\lambda)]$.
- **Disavow**: is an interactive protocol between $c_i$ and $\nu'$ with common input $(m, \sigma, pk_i, pck, pck_i)$. The output is $b \in \{\text{Accept}, \bot\}$. The protocol must be both complete and sound. For completeness, we require that there is some $c_i$ such that if $\sigma$ is an invalid ADCS on $m$ then $b = \text{Accept}$. For soundness, we require that for all $c_i$ if $\sigma$ is a valid ADCS on $m$, then $\text{Prob}[(\text{Disavow}(m, \sigma, pk_i, pck, pck_i) = \text{Accept}) < \text{negl}(\lambda)]$.

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 confirmer. 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 \( A \) has a non-negligible advantage in the following game:

**Game S**

1. The challenger \( B \) takes as input a security parameter \( \lambda \) and gives \((pks, pk_1, ..., pk_n, pck)\) to \( A \).
2. The adversary \( A \) is permitted to a series of queries:
   - **ConfirmedSign queries:** \( A \) submits a message \( m \) and receives an ADCS \( \sigma \) on \( m \).
   - **Extract queries:** \( A \) receives a secret confirmation key \( sck \).
   - **Designate queries:** \( A \) submits \( C_i \)'s public key \( pk_i \) and receives a corresponding public confirmation key \( pck_i \).
   - **Confirm(\( C_i \), pk)\)/Disavow(\( C_i \), \( \sigma \)) queries:** \( A \) executes Confirm/Disavow protocol in the confirmor role.
   - **Confirm(\( C_i \), pk)\)/Disavow queries:** \( A \) executes Confirm/Disavow protocol in the verifier role.
   - **Corrupt\( _j \) queries:** \( A \) submits a confirmor's public key \( pk \) and receives a corresponding secret key \( sk \).
3. At the end of this game, \( A \) outputs a pair \((m', \sigma')\).

\( A \) wins the game if \( \text{Conf}(\sigma, m, \sigma, \sigma') \) = Accept such that \( m' \) has never been queried to the ConfirmedSign oracle and that \((m', \sigma')\) has never been accepted at the Confirm(\( C_i \)) queries earlier. The \( A \)'s advantage \( \text{Adv}^S(\lambda) \) is defined to be probability that \( A \) 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 \( \sigma \) may have already been produced by legal signer). In an ADCS scheme, the "legal" confirmers should know both the confirmor'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 \( A \) has a non-negligible advantage in both of Game C-1 and Game C-2.

**Game C-1**

1. The adversary \( A \) classifies the candidates for confirmers \((C_1, ..., C_n)\) into \( C_i(i = 1, ..., l) \in SC_h \) and \( C_j(j = l + 1, ..., n) \in SC_c \), and notifies the challenger \( B \).
2. \( B \) takes as input a security parameter \( \lambda \) and gives \((pks, pk_1, ..., pk_n, pck)\) to \( A \).
3. \( A \) is permitted to a series of queries:
   - **ConfirmedSign queries:** \( A \) submits a message \( m \) and receives an ADCS \( \sigma \) on \( m \).
   - **Extract queries:** \( A \) receives a secret confirmation key \( sck \).
   - **Designate queries:** \( A \) submits \( C_i \)'s public key \( pk_i \) and receives a corresponding public confirmation key \( pck_i \). Note that a query of \( pk_i(C_j) \in SC_c \) is prohibited.
   - **Confirm(\( C_i \), \( \sigma \))/Disavow(\( C_i \), \( \sigma \)) queries:** \( A \) executes Confirm/Disavow protocol in the confirmor role.
   - **Confirm(\( C_i \), \( \sigma \))/Disavow queries:** \( A \) executes Confirm/Disavow protocol in the verifier role.
   - **Corrupt\( _j \) queries:** \( A \) submits a confirmor's public key \( pk \) and receives a corresponding secret key \( sk \).
4. At the end of this game, \( A \) outputs a pair \((m, \sigma, pk_i, pck_i)\).

\( A \) wins the game if \( \text{Conf}(\sigma, m, \sigma, pk_i, pck_i) \) = Accept. The \( A \)'s advantage \( \text{Adv}^C(\lambda) \) is defined to be probability that \( A \) wins this game.

**Game C-2**

1. The adversary \( A \) classifies the candidates for confirmers \((C_1, ..., C_n)\) into \( C_i(i = 1, ..., l) \in SC_h \) and \( C_j(j = l + 1, ..., n) \in SC_c \), and notifies the challenger \( B \).
2. \( B \) takes as input a security parameter \( \lambda \) and gives \((pks, pk_1, ..., pk_n, pck)\) to \( A \).
3. \( A \) is permitted to a series of queries:
ON THE SECURITY OF ADDING CONFIRMERS INTO DESIGNATED CONFIRMER SIGNATURES

Confirmsign queries: A submits a message m and receives an ADCS S on m.

Designate queries: A submits C 1 ∊ SC’s public key pk and receives a corresponding public confirmation key pk c for C 1 . Note that a query of pk i ∊ SC is prohibited.

Confirmsign(query; C 1 )/Disavow(query; C 1 ) queries: A executes Confirmsign/Disavow protocol in the verifier role.

Confirmsign(query; C 1 )/Disavow(query; C 1 ) queries: A executes Confirmsign/Disavow protocol in the verifier role with C j ∊ SC. It is prohibited that A executes Confirmsign/Disavow protocol with C j ∊ SC.

Corrupt; queries: A submits a signer S’s verification key pk and receives a corresponding signing key sk S.

Corrupt; queries: A submits a confirmer C j ∊ SC’s public key pk j and receives a corresponding secret key sk j . Note that a query of pk i ∊ SC is prohibited.

4 At the end of this game, A outputs a pair (m, σ, pk i , pk c i ) .

A wins the game if Confirmsign(query; C 1 ) (m, σ, pk i , pk c i ) = Accept. The A’s advantage Adv AC−2 (A) is defined to be probability that 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 G be a bilinear group where |G| = p for some prime p. Let g be a generator of G.

KeyGen: pick random x, y ∊ Z p* and compute u ← g x , v ← 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 ∈ Z p* , pick a random r ∈ Z p* and compute \( s \leftarrow g^{(x+my+r)} \). 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 \( \sigma = (s, r) \).

Verify: given a verification key (u, v), a message m, and a signature \( \sigma = (s, r) \), verify that 
\[
e(s, s) = e(uv^m g^r, g).
\]

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

Theorem 1. Suppose the \( (k + 1/t', \epsilon') \)-square roots assumption holds in \((G, G_1)\). Then the ZCSM06 signature scheme is \((t, q_5, \epsilon)\)-secure against existential forgery under a chosen message attack provided that 
\[
q_5 \leq k + 1, \epsilon = 2\epsilon' + 4q_5/p \approx 2\epsilon', t \leq t' - \Theta(q_5 T).
\]

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 Confirmsign algorithm is similar to ZCW08 (Zhang et al., 2008).

Let G be a bilinear group where |G| = p for some prime p. Let g be a generator of G.

KeyGen: pick random x, y ∊ Z p* , random k ∊ Z p* such that k is a quadratic residue modulo p and compute \( u \leftarrow g^x, v \leftarrow g^y, K = g^k \) and \( b \leftarrow g^{12} \). \( (sk, pk) = (sk, pk) = (b, K) \).

Pick random \( a_i \in Z_p^* \) (\( i = 1, \ldots, n \)) and compute \( A_i \leftarrow g^{b_i} \) (\( i = 1, \ldots, n \)). \( (sk, pk) = (a_i, A_i) \).

Sign: is same as the ZCSM06 signature scheme.

Verify: is same as the ZCSM06 signature scheme.

Confirmsign: given \( (x, y) \), a message m ∊ Z p* , pick a random r ∊ Z p* and compute s = \( K^{(x+my+r)^{1/2}} \). 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 \( sk = b \). Once b is revealed, everyone can verify the signature \( \sigma \) 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 \( sk = b \) and \( sk_i = a_i \), computes \( B_i = b^{1/a_i} \), and discloses \( pk_i \) as the public confirmation key for \( C_i \). Any verifier \( v' \) can verify the validity of \( B_i \) by 
\[
e(K, K) = e(A_i, B_i).
\]

Confirm: given a pair \( (m, \sigma) \), \( C_i \) verifies an ADCS by 
\[
e(s, s) = e(uv^m g^r, B_i)^{a_i} \land e(g, g)^{a_i} = e(A_i, g)
\]

otherwise, \( C_i \) outputs ⊥.
Disavow: given a pair \((m, \sigma), C_i\) verify an ADCS by
\[ e(s, s) = e(uvm^g, B_i)^{m_i}. \]
If the equality does not hold, \(C_i\) execute the interactive ZKPoK protocol with \(\Psi\) as follows;
\[ PK\{(a_i) : e(s, s) \neq e(uvm^g, B_i)^{a_i} \land e(g, g)^{a_i} = e(A_i, g)\}. \]
otherwise, \(C_i\) 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 \((t', q_p, e, \epsilon)\)-secure against existential forgery under a chosen message attack. Then the proposed scheme is \((t, q, e)\)-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\((A, \psi')\), Disavow\((A, \psi')\), Confirm\((C, \psi)\), Disavow\((C, \psi)\) queries.

Proof. Let \(A\) be a PPT adversary that has negligible advantage \(Adv^S(\mathcal{A})\). 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 \((g, u = g^r, v = g^s)\) generated by ZCSM06’s KeyGen algorithm. \(\mathcal{B}\) picks a random \(k \in \mathbb{Z}_p^\ast\) and compute \(K = g^k, b = g^s\). \(\mathcal{B}\) also picks a random \(a_i \in \mathbb{Z}_p^\ast\) for \(i = 1, ..., n\). \(\mathcal{A}\) is given \((g, u, v, K, A_i)(i = 1, ..., n)\), \(\mathcal{A}\) 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 \(\sigma = (s, r)\), computes \(s \leftarrow r\), and returns \(\sigma = (s, r)\) to \(\mathcal{A}\).

When \(\mathcal{A}\) makes a Confirm\((A, \psi')\) / Disavow\((A, \psi')\) 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\((C, \psi)\) / Disavow\((C, \psi)\) 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 \(C_i\), \(\mathcal{B}\) returns \(B_i = b^{1/a_i}\) to \(\mathcal{A}\).

When \(\mathcal{A}\) makes an Extract query, \(\mathcal{B}\) returns the secret confirmation key \(b\) to \(\mathcal{A}\).

When \(\mathcal{A}\) makes a Corrupt\(_c\) query for a confirmer \(C_i\), \(\mathcal{B}\) returns \(a_i\) to \(\mathcal{A}\).

\(\mathcal{B}\) does not abort during above simulation, and finally \(\mathcal{A}\) outputs \((m', s', r')\) such that \(e(s', s') = e(uvm^g, g^r)\). \(\mathcal{B}\) succeeds in forgery on ZCSM06 signature scheme.

Theorem 3. Suppose the \((t', \epsilon, 3)\)-BDHE assumption holds in \(\mathbb{G}\). Then the proposed scheme is \((t, q, \epsilon)\)-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 \(Adv^C(\mathcal{A}) \geq \epsilon\), there exists an algorithm which solves \((t', \epsilon, 3)\)-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\((A, \psi')\), Disavow\((A, \psi')\), Confirm\((C, \psi)\), Disavow\((C, \psi)\) queries.

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

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

\(\mathcal{A}\) outputs the list of \(C_i \in SC_h\) (the identity of selective honest confirmer) and \(C_j \in SC_c\) (the identity of selective corrupted confirmer) to \(\mathcal{B}\) and notifies \(\mathcal{B}\).

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

- For \(C_i \in SC_h\), \(\mathcal{B}\) picks random values \(a_i(i = 1, ..., l)\) as secret keys and computes \(A_i = (g^{e_j})^{a_i}(i = 1, ..., l)\) as public keys.
- For \(C_j \in SC_c\), \(\mathcal{B}\) picks random values \(a_j(j = l + 1, ..., n)\) as secret keys and computes \(A_j = g^{e_j}(j = l + 1, ..., n)\) as public keys.

\(\mathcal{A}\) is given the \((u, v), (A_1, ..., A_l), (A_{l+1}, ..., A_n), K\). \(\mathcal{A}\) 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 \(r \in \mathbb{Z}_p^\ast\) and computes \(s \leftarrow K^{(x + my + r)^{1/2}}\). Then \(\mathcal{B}\) returns \(\sigma = (s, r)\) to \(\mathcal{A}\).
When $A$ makes a $\text{Confirm}_{(A',v')}/\text{Disavow}_{(A',v')}$ query, $B$ performs the Confirm/Disavow protocol in the verifier role. $B$ need not know any secret information.

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

When $A$ makes a Designate query for a confirmer $C_i$, $B$ computes $B_i \leftarrow (G^2)^{i/2} = b^{i/(2m)}$, and returns $B_i$, to $A$.

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

When $A$ makes a Corrupt query, $B$ returns $x,y$ to $A$.

When $A$ makes a Confirm query for $C_j \in SC_c$, $B$ returns $a_j \cdot A_i$. The query for a confirmer $C_i \in SC_h$ is prohibited.

$B$ does not abort during above simulation, and finally output a pair $(m^*, s^*, r^*, A_r, B_r)$ which is accepted in Confirm protocol. Here, simulator $B$ can obtain $\log A_r = z \cdot r$, by using the knowledge extractor and can get $t$ (because $B$ generated $a_t$). $B$ computes $G^t = (G^2)^{t/2}$, then outputs $e(G^{t}, H) = e(G, H)^{t/2}$ that is, $B$ solves 3-BDHE problem.

**Lemma 2.** If there exists a t-time algorithm $A$ which satisfies $\text{Adv}^{\text{H}}(A) \geq \epsilon$, there exists an algorithm which solves $(t', \epsilon, 3)$-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 $G'$. \text{or ConfirmSign} or ConfirmSign$(A',v')/\text{Disavow}(A',v')$ or $\text{Confirm}_{(C,A'}/\text{Disavow}_{(C,A')}$ queries.

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

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

$B$ outputs the list of $C_i \in SC_h$ (the identity of selective honest confirmers) and $C_j \in SC_c$ (the identity of selective corrupted confirmers) and notifies $B$.

$B$ 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. $B$ sets $K = g^z$ as a public confirmation key. Furthermore, $B$ generates confirmers’ public and secret key pairs as follows:

- For $C_i \in SC_h$, $B$ picks random values $a_i (i = 1, \ldots, l)$ and computes $A_i \leftarrow (g^2)^{a_i} (i = 1, \ldots, l)$ as public keys.
- For $C_j \in SC_c$, $B$ picks random values $a_j (j = l + 1, \ldots, n)$ and computes $A_j \leftarrow g^{a_j} (j = l + 1, \ldots, n)$ as secret keys and computes $A_j = g^{a_j}$ as $A_j$.

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

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

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

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

When $A$ makes a Designate query for a confirmer $C_i \in SC_h$, $B$ computes $B_i \leftarrow (g^{z})^{1/(2m)}$ and returns $B_i$ to $A$. The query for a confirmer $C_i \in SC_c$ is prohibited.

When $A$ makes a Corrupt query, $B$ returns $x, y$ to $A$.

When $A$ makes a Confirm query for $C_j \in SC_c$, $B$ returns $a_j \cdot A_i$. The query for a confirmer $C_i \in SC_h$ is prohibited.

$B$ does not abort during above simulation. Finally, $A$ output a pair $(m^*, s^*, r^*, A_r, B_r)$ which is accepted in Confirm protocol. Note that the equation $e(K, K) = e(A_r, B_r)$ holds.

Here, simulator $B$ can obtain $\log A_r$, using the knowledge extractor. Let $\alpha_1$, be $\log A_r$. $e(K, K) = e(g, g^{2})$ holds. On the other hand, $e(A_r, B_r) = e(g, B_r^{2})$ holds.

Hence, $B$ gets $B_r^{2} = g^{2}$ and computes $e(g, h) = e(g, h)^2$. So, $B$ solves 3-BDHE problem.

### 6 CONCLUSIONS

In this paper, we have shown new designated conifer 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. $SC_h$ and $SC_c$ 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


