New \((t, n)\) threshold directed signature scheme with provable security

Rongxing Lu \(^a, *\), Xiaodong Lin \(^b\), Zhenfu Cao \(^a\), Jun Shao \(^a\), Xiaohui Liang \(^a\)

\(^a\) Department of Computer Sciences and Engineering, Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai, China

\(^b\) Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

Received 8 August 2006; received in revised form 6 July 2007; accepted 29 July 2007

Abstract

Directed signature scheme allows only a designated verifier to check the validity of the signature issued to him; and at the time of trouble or if necessary, any third party can verify the signature with the help of the signer or the designated verifier as well. Due to its merits, directed signature scheme is widely used in situations where the receiver’s privacy should be protected. Threshold directed signature is an extension of the standard directed signature, in which several signers may be required to cooperatively sign messages for sharing the responsibility and authority. To the best of our knowledge, threshold directed signature has not been well studied till now. Therefore, in this paper, we would like to formalize the threshold directed signature and its security model, then present a new \((t, n)\) threshold directed signature scheme from bilinear pairings and use the techniques from provable security to analyze its security.

© 2007 Elsevier Inc. All rights reserved.

Keywords: Directed signature; Threshold directed signature; Bilinear pairings; Provable security

1. Introduction

Digital signature, the electronic version of handwritten signature, is one of the most important techniques in modern information security system for its functionality of providing data integrity and authentication. However, due to its non-repudiable and public-verifiable properties, it is not suitable for applications where messages signed are personally or commercially sensitive to the receiver, such as a bill of tax, a bill of health, etc. Therefore, to prevent potential misuse of signatures, it is preferable to modify some constraints on non-repudiable and public-verifiable properties. Up to now, several variants of digital signature have been proposed to protect user’s privacy (e.g. in [6,7,5,14]). Undeniable signature was first introduced at Crypto’89 by Chaum [6]. Such signature requires the signer’s cooperation each time in verification. However, one obvious problem behind this idea is that when the signer becomes unavailable or corrupted, nothing can be
determined. As an alternate approach, Chaum also introduced the designated confirmer signature at Eurocrypt’94 [5]. In this scheme, a trusted third party called confirmer was introduced in order to protect the signer from a coercer, and the verification of such a signature requires the assistance of the confirmer. Therefore, we can easily see that, in both of these two schemes, the receivers cannot verify the validity of signatures only by themselves.

However, this is not the case in some applications. There are many situations where interactive proofs are not always available and the designated signature receiver can be predetermined. Therefore, Lim and Lee [17] first proposed the concept of directed signature at Auscrypt’92 to solve above problem. In a directed signature scheme, after a signer sends a signed message $m$ to a designated verifier, only the designated verifier can directly verify the signature on message $m$ while the others know nothing on the origin and validity of the message $m$ without the help of the signer or the designated verifier. Due to this property, directed signature is desirable in many situations where message signed is personally or commercially sensitive to and/or somewhat obligatory on the designated receiver. In addition, at the time of trouble or if necessary, both the signer and the designated verifier in a directed signature scheme can prove to a third party that the signature is valid.

Lim and Lee [17] proposed the first directed signature scheme based on Guillou–Quisquater scheme [11], but it lacks formal definitions and proofs of security. At Asiacrypt’05, along the research line, Laguillaumie et al. [18] deeply studied the universally convertible directed signatures and provided the security proofs in the random oracle model [4]. Their work is very pretty. However, to the best of our knowledge, researches in threshold directed signature are still not satisfactory. Like most threshold schemes [8,9,24,13,12,23,15,16], threshold directed signature is an extension of the standard directed signature, in which several signers may be required to cooperatively sign messages for sharing the responsibility and authority. More precisely, properties of a typical $(t,n)$ threshold directed signature scheme are described as follows.

1. A signer group is formed by $n$ signers. Any $t$ or more signers in the group can cooperate to generate a valid group signature for a designated verifier, while fewer than $t$ signers cannot.
2. Only the designated verifier can directly verify the group signature with only knowing the public key associated to the signer group. It is not necessary to perceive all individual public keys of those who generate the signature, which preserves the signer anonymity.
3. At the time of trouble or if necessary, both the signer group and the designated verifier can prove to a third party that the group signature is valid under the signer group’s public key.

Motivated by the mentioned above, we focus ourselves on threshold directed signature scheme in this paper. Concretely, we regard the main contribution of this paper to be two-fold significance: (i) We formalize the definition for a $(t,n)$ threshold directed signature and propose its security model that captures the stronger notion of unforgeability. (ii) We present a new $(t,n)$ threshold directed signature scheme from bilinear pairings, and use the techniques from provable security to analyze its security [20,21].

The rest of this paper is organized as follows. In the next section, we set up the formal definition and security model for threshold directed signature. In Section 3, we review the bilinear pairings and complexity assumptions on which we build. Then, we propose our new $(t,n)$ threshold directed signature scheme in Section 4, followed by the security analysis and performance discussion in Sections 5 and 6, respectively. Finally, we draw our conclusions in Section 7.

2. Definitions

2.1. Notations

We let $\mathbb{N} = \{1, 2, 3, \ldots\}$ be the set of positive integers. If $x$ is a string, then $|x|$ denotes its length, while if $S$ is a set then $|S|$ denotes its size. If $k \in \mathbb{N}$ then $1^k$ denotes the string $k$ ones. If $S$ is a set then $s \leftarrow S$ denotes the operation of picking a random element $s$ of $S$ uniformly. Unless otherwise indicated, we denote that $\mathbb{A} = \{A_1, A_2, \ldots, A_n\}$ as the signer group, which consists of $n$ signers $A_1, A_2, \ldots, A_n$, $B$ as the designated verifier and $C$ as any third party in the following scheme.
2.2. Definitions for threshold directed signature

**Definition 1 (Threshold Directed Signature).** A \((t,n)\) threshold directed signature scheme \(\mathcal{TDS}\) consists of the following five algorithms \{KGen,KShare,TSign,DVerify,PVerify\}:

1. Key generation algorithm (KGen): On input an unary string \(1^k\) where \(k\) is a security parameter, this algorithm outputs the public and private key pair \((pk_{A_1},sk_{A_1}), (pk_{A_2},sk_{A_2}), \ldots, (pk_{A_n},sk_{A_n}), (pk_B,sk_B)\) of \(n\) signers \(A_1, A_2, \ldots, A_n\) and the designated verifier \(B\). KGen is a probabilistic polynomial-time algorithm.

2. Key sharing algorithm (KShare): To setup a \((t,n)\) threshold signer group \(A = \{A_1,A_2,\ldots,A_n\}\), where \(n \geq 2t - 1\), \(n\) signers \(A_i (1 \leq i \leq n)\) jointly execute the verifiable secret sharing operations. In the end, each \(A_i \in A\) holds another shadow public and private key pair \((pk'_{A_i},sk'_{A_i})\), and the signer group’s public key \(pk_A\) is the aggregation of each signer’s public key.

3. Threshold signing algorithm (TSign): To sign a message \(m\) for the designated verifier \(B\), \(t\) or more out of \(n\) signers in group \(A\), with their individual private keys, outputs the directed signature \(\sigma\) on \(m\).

4. Directed verifying algorithm (DVerify): On input of a purported signature \(\sigma\) of message \(m\), the designated verifier \(B\)’s private key \(sk_B\) and the signer group’s public key \(pk_A\), outputs “1” if \(\sigma\) is valid, and “0” otherwise.

5. Public verifying algorithm (PVerify): On input of a purported signature \(\sigma\) of message \(m\), the signer group \(A\)’s public key \(pk_A\) and the designated verifier \(B\)’s public key \(pk_B\), a third party \(C\), with a verifiable Aid provided by the signer group \(A\) or the designated verifier \(B\), outputs “1” if \(\sigma\) is valid, and “0” otherwise.

2.3. Security notions for threshold directed signature

**Security against existential forgery under chosen message attack.** For digital signatures, the well-known strong security notion is existential forgery against adaptive chosen message attacks (EF-CMA) presented by Goldwasser et al. [10]. Therefore, with respect to the unforgeability of \((t,n)\) threshold directed signature schemes, we will define it alone the same lines. In the random oracle model [4], we consider the most powerful adversary \(\mathcal{A}\) as follows.

1. The adversary \(\mathcal{A}\) can corrupt and control the designated verifier \(B\) and at most \(t - 1\) signers \((A_1^*,A_2^*,\ldots,A_{t-1}^*)\) in signer group \(A\).

2. The adversary \(\mathcal{A}\) is also allowed to access to the signing oracle \(\mathcal{O}_S\) and the random oracle \(\mathcal{O}_H\).

3. In the end, the adversary \(\mathcal{A}\) returns a new valid signature \(\sigma^*\) on message \(m^*\). There is a natural restriction that the signature \(\sigma^*\) has not been obtained from the signing oracle \(\mathcal{O}_S\) before.

**Definition 2 (Unforgeability).** Let \(\mathcal{TDS}\) be a threshold directed signature scheme, let \(\mathcal{A}\) be an EF-CMA adversary against \(\mathcal{TDS}\). We consider the following random experiments, where \(k\) is the security parameter:

\[
\begin{align*}
\text{Experiment} & \text{Exp}_{\mathcal{TDS},\mathcal{A}}^{\text{EF-CMA}}(k) \\
(pk_{A_1},sk_{A_1}),\ldots,(pk_{A_n},sk_{A_n}),&(pk_B,sk_B) \leftarrow \text{KGen}(k), \\
(pk'_{A_1},sk'_{A_1}),\ldots,(pk'_{A_n},sk'_{A_n}),&(pk_A \leftarrow \text{KShare}((pk_{A_1},sk_{A_1}),\ldots,(pk_{A_n},sk_{A_n}))), \\
(\sigma^*,m^*) & \leftarrow \mathcal{A}_H^C,\mathcal{O}_S(\text{corrupt}A_1^*,\ldots,A_{t-1}^*,B) \\
\text{Return}& \text{DVerify}(pk_{A_1},pk_B,sk_B,\sigma^*,m^*)
\end{align*}
\]

We then define the success probability of \(\mathcal{A}\) via

\[
\text{Succ}_{\mathcal{TDS},\mathcal{A}}^{\text{EF-CMA}}(k) = \Pr[\text{Exp}_{\mathcal{TDS},\mathcal{A}}^{\text{EF-CMA}}(k) = 1]
\]

Let \(t \in \mathbb{N}\) and \(\epsilon \in [0,1]\). We say that the \(\mathcal{TDS}\) is \((\tau, \epsilon)\)-secure if no EF-CMA adversary \(\mathcal{A}\) running in time \(\tau\) has a success \(\text{Succ}_{\mathcal{TDS},\mathcal{A}}^{\text{EF-CMA}}(k) \geq \epsilon\).
Verifiable directedness. The property of verifiable directedness is another requirement on threshold directed signature scheme. When we design a TDS scheme, we should at least computationally rule out any signature distinguisher \( \mathcal{D} \) in a scheme. As usual, the Invisibility [7] and Transitivity of a TDS scheme should be considered. In the following, we informally define the property of verifiable directedness.

**Definition 3 (Verifiable Directedness).** A threshold directed signature scheme is said to be with the property of verifiable directedness, the following two conditions should be satisfied:

1. **Invisibility:** We use the following game to define the notion of signature Invisibility for a directed signature scheme. Let \( \mathcal{D} \) be a (probabilistic polynomial time) distinguisher and \( \mathcal{CH} \) is a challenger. We consider the distinguisher \( \mathcal{D} \) runs in two stages. In the find stage, \( \mathcal{D} \) takes as input the signer group \( A \)'s public key \( pk_A \) and the designated verifier \( B \)'s public key \( pk_B \), and outputs two same length messages \( m_0 \) and \( m_1 \). Then, in the guess stage, \( \mathcal{D} \) takes as input a valid challenge signature \( \sigma \) from \( \mathcal{CH} \) formed by signing the message \( m_b \), where \( b \in \{0,1\} \). In the end, \( \mathcal{D} \) returns his guess \( b' = \{0,1\} \). The distinguisher \( \mathcal{D} \) wins the game if \( b' = b \). We define the advantage of \( \mathcal{D} \) as:

\[
\text{Adv}_{TDS} = 2\Pr[b' = b] - 1
\]

where \( \Pr[b' = b] \) denotes the probability that \( b' = b \). If the advantage \( \text{Adv}_{TDS} \) of \( \mathcal{D} \) is negligible, we say Invisibility is satisfied.

2. **Transitivity:** with the help of the signer group \( A \) or the designated verifier \( B \), any third party \( C \) can check the validity of a signature.

3. Bilinear pairings and complexity assumptions

Bilinear pairing is an important cryptographic primitive and has been widely adopted in many positive applications in cryptography [2,3]. Let \( G_1 \) be a cyclic additive group and \( G_2 \) be a cyclic multiplicative group of the same prime order \( q \). We assume that the discrete logarithm problems in both \( G_1 \) and \( G_2 \) are hard. A bilinear pairing is a map \( e : G_1 \times G_1 \rightarrow G_2 \) which satisfies the following properties:

1. **Bilinearity:** For any \( P, Q \in G_1 \) and \( a, b \in \mathbb{Z}_q \), we have \( e(aP, bQ) = e(P, Q)^{ab} \).
2. **Non-degeneracy:** There exists \( P \in G_1 \) and \( Q \in G_1 \) such that \( e(P, Q) \neq 1_{G_2} \).
3. **Computability:** There exists an efficient algorithm to compute \( e(P, Q) \) for all \( P, Q \in G_1 \).

From the literature [2], we note that such a bilinear pairing may be realized using the modified Weil pairing associated with supersingular elliptic curve. Next the related complexity assumptions are as follows.

**Definition 4 (Computational Diffie–Hellman (CDH) Problem).** For \( a, b \in \mathbb{Z}_q^* \), given \( P, aP, bP \in G_1 \), compute \( abP \in G_1 \). An algorithm \( \mathcal{A} \) is said to solve the CDH problem with an advantage \( \epsilon \) if

\[
\text{Adv}_{G_1}^{\text{CDH}}(\mathcal{A}) = \Pr[\mathcal{A}(P, aP, bP) = abP] \geq \epsilon
\]

where the probability \( \epsilon \) is taken over the random choices of \( a \) and \( b \) that \( \mathcal{A} \) makes .

**Definition 5 (Decisional Diffie–Hellman (DDH) Problem).** For \( a, b, c \in \mathbb{Z}_q^* \), given \( P, aP, bP, cP \in G_1 \), decide whether \( c = ab \in \mathbb{Z}_q \). The DDH problem is easy in \( G_1 \), since we can compute \( e(aP, bP) = e(P, P)^{ab} \) and decide whether \( e(P, P)^{ab} = e(P, P)^c \) [2].

**Definition 6 (Gap Diffie–Hellman (GDH) Group).** If the DDH problem can be solved in polynomial time, while no polynomial algorithm can solve the CDH problem with non-negligible advantage within polynomial time, then \( G_1 \) is referred to as a Gap Diffie–Hellman group.

**Definition 7 (Inverse CDH (ICDH) Problem).** For \( a, b \in \mathbb{Z}_q^* \), given \( P, abP, bP \in G_1 \), compute \( aP \in G_1 \). An algorithm \( \mathcal{A} \) is said to solve the ICDH problem with an advantage of \( \epsilon \) if
The CDH problem is equivalent to the ICDH problem in \( G_1 \).

In 2003, Bao et al. [1] discussed some variations of the Diffie–Hellman problems and proved that the CDH problem is equivalent to the ICDH problem. Thus, the reader may refer to [1] for details of proof. Throughout this paper, we assume that the CDH problem is intractable in \( G_1 \), which means there is no polynomial time algorithm to solve the CDH problem and ICDH problem with non-negligible probability.

4. New \((t,n)\) threshold directed signature scheme

In this section, we present our new \((t,n)\) threshold directed signature scheme, which consists of the following algorithms.

KGen: Given a security parameter \( k \). Let \( G_1 \) be a GDH group of prime order \( q \), where \( |q| = k \), and \( P \) be a generator of \( G_1 \). Let \( G_2 \) be a cyclic multiplicative group of the same prime order \( q \). Then, the bilinear pairing \( e : G_1 \times G_1 \rightarrow G_2 \) is given, and a secure cryptographic hash function \( H : \{0, 1\}^* \rightarrow G_1 \) is also selected. In the end, the system parameters and their descriptions are \( \{G_1, G_2, q, e, P, H\} \).

Each signer \( A_i \in A \), \( 1 \leq i \leq n \), chooses a random number \( x_{A_i} \leftarrow \mathbb{Z}_q^* \) as his private key and publishes the corresponding public key \( Y_{A_i} = x_{A_i}P \).

Similarly, the designated verifier \( B \) also chooses a random number \( x_B \leftarrow \mathbb{Z}_q^* \) as his private key and publishes his public key \( Y_B = x_BP \).

KShare: To generate secret shares in the signer group \( A = \{A_1, A_2, \ldots, A_n\} \), the group \( A \) applies a \((t,n)\) verifiable secret sharing scheme [19,22] as follows.

1. Each \( A_i \in A \) randomly chooses a \( t-1 \) degree polynomial:

\[
f_i(x) = x_{A_i} + b_{i,1}x + \ldots + b_{i,t-1}x^{t-1} \mod q
\]

where \( x_{A_i} \) is \( A_i \)'s private key. \( A_i \) then broadcasts \( D_{i,l} = b_{i,l}P \), \( l = 1, 2, \ldots, t-1 \) in signer group \( A \).

2. Each \( A_i \in A \) computes and sends \( f_i(j) \) to \( A_j \in A \) via a secure channel for \( j \neq i \).

3. After receiving \( f'_j(i) \) from \( A_j \), \( A_i \) can validate it by checking

\[
f'_j(i) = Y_{A_i} + \sum_{l=1}^{t-1} i^l D_{j,l}.
\]

4. Let \( f'(x) = \sum_{i=1}^{n} f'_i(x) \mod q \), then \( A_i \)'s secret shadow is \( f'(i) \). \( A_i \) computes and broadcasts \( Y'_{A_i} = f'(i)P \) in the signer group \( A \).

5. In the end, the signer group \( A \)'s public key \( pk_A \) is the sum of all signers’ public keys as follows,

\[
Y_A = f'(0)P = \sum_{i=1}^{n} f'_i(0)P = \sum_{i=1}^{n} x_{A_i}P = \sum_{i=1}^{n} Y_{A_i}.
\]

TSign: Suppose that the signer group \( A \) wants to make a signature on message \( m \in \{0,1\}^* \) for the designated verifier \( B \), i.e. only \( B \) can directly verify the signature. Without loss of generality, here we assume that \( TG = \{A_1, A_2, \ldots, A_t\} \in A \) are the actual signers, and \( A_1 \) is a designated dealer who is chosen from \( TG \) in advance. Then, they proceed as follows.

1. First, the dealer \( A_1 \) chooses a random number \( r \leftarrow \mathbb{Z}_q^* \), and computes

\[
R = rY_B
\]

Then, \( A_1 \) computes \( \beta_1 \) where
\[ \beta_i = f'(i) \prod_{j=2}^{t} \frac{0-j}{1-j} \cdot H(R|m) \]  

(5)

In the end, \( A_1 \) broadcasts \( H(R|m) \) in \( TG \).

2. Each \( A_i \in TG/\{A_1\} \) uses \( f'(i) \) to compute

\[ \beta_i = f'(i) \prod_{j=1, j \neq i}^{t} \frac{0-j}{1-j} \cdot H(R|m) \]  

(6)

and sends \( \beta_i \) back to the dealer \( A_1 \).

3. After receiving \( \beta_i \) from \( A_i \in TG \), the \( A_1 \) can validate it by checking

\[ e(\beta_i, P) = e\left( \prod_{j=1, j \neq i}^{t} \frac{0-j}{1-j} \cdot H(R|m), Y_{A_i} \right) \]  

(7)

If it holds, the individual signature \( \beta_i \) is accepted, otherwise rejected.

4. If all individual signatures \( \beta_i \), \( 1 \leq i \leq t \), are valid, the dealer \( A_1 \) then computes

\[ S = \sum_{i=1}^{t} \beta_i - rP = \sum_{i=1}^{t} \left( f'(i) \prod_{j=1, j \neq i}^{t} \frac{0-j}{1-j} \cdot H(R|m) \right) - rP = f'(0)H(R|m) - rP \]  

(8)

5. In the end, the signature on message \( m \) is \( \sigma = (R, S) \).

DVerify: Upon receiving \( \sigma = (R, S) \), the designated verifier \( B \) first uses his private key \( x_B \) to check the following equality

\[ e\left( S + \frac{1}{x_B} R, P \right) = e(H(R|m), Y_A) \]  

(9)

If it does hold, the signature \( \sigma = (R, S) \) will be accepted, otherwise, rejected. Since

\[ e\left( S + \frac{1}{x_B} R, P \right) = e\left( S + \frac{1}{x_B} rY_B, P \right) \]

\[ = e(S + rP, P) = e(f'(0)H(R|m) - rP + rP, P) \]

\[ = e(f'(0)H(R|m), P) = e(H(R|m), f'(0)P) \]

\[ = e(H(R|m), Y_A) \]

PVerify: In time of trouble or if necessary, either the dealer \( A_1 \) on behalf of the signer group \( A \) or the designated verifier \( B \) provides an AID \( R' = rP \) to a third party \( C \), so \( C \) is able to verify the signature \( \sigma = (R, S) \) as follows.

1. First, \( C \) checks the validity of \( R' \) by the equality

\[ e(R', Y_B) = e(R, P) \]  

(10)

If it holds, \( R' = r \cdot P \) will be accepted, otherwise rejected, since

\[ e(R', Y_B) = e(rP, x_B P) \]

\[ = e(rY_B, P) = e(R, P) \]

2. Then, with valid \( R' \), the third party \( C \) can verify the signature \( \sigma = (R, S) \) by equality

\[ e(S + R', P) = e(H(R|m), Y_A) \]  

(11)

Clearly, the correctness of the \( TDF \) scheme is straightforward and can be easily verified.
5. Security analysis

In this section, we study the security of the new \( \mathcal{F} \mathcal{H} \mathcal{S} \) scheme, and the security results are described in the following theorems.

**Theorem 2.** Let \( \mathcal{A} \) be an adversary who can produce, with success probability \( \text{Succ}^{\text{EF-CMA}}_{\text{TDS}, \mathcal{A}} = \epsilon \), an existential forgery signature of the proposed \( (t, n) \mathcal{F} \mathcal{H} \mathcal{S} \) scheme, under chosen-message attacks within time \( \tau \), after \( q_h \) and \( q_s \) queries to the hash function \( H \) and the signing oracle, respectively. Then the CDH problem in \( G_1 \) can be solved with another probability \( \epsilon' \) within time \( \tau' \), where

\[
\epsilon' \geq \frac{(t - 1)!(n - t + 1)!}{n!} \cdot \frac{1}{q} \cdot \frac{q_h \cdot q_s}{q};
\]

\[
\tau' \leq \tau + (nt - t^2 + 3t + 1 + q_h + 4q_s)T_{\text{smul}};
\]

with \( T_{\text{smul}} \) denotes the time for a scale multiplication operation in \( G_1 \).

**Proof.** We now prove the theorem based on the formalism introduced by Shoup [20,21]. Assume the \( \mathcal{A} \) is the adversary. We define a sequence of games \( \text{Game}_0, \text{Game}_1, \ldots \), of modified attack games starting from the actual game \( \text{Game}_0 \). Then, with these incremental games, we reduce a CDH problem instance (randomly choose \( a, b \in \mathbb{Z}_q^* \), given \( P, aP, bP \in G_1 \), compute \( abP \)) to an attack against the threshold directed signature. We show that the adversary \( \mathcal{A} \) can help us to solve the CDH problem in \( G_1 \).

**Game_0:** This is an actual game, in the random oracle model [4]. The adversary \( \mathcal{A} \) is allowed to access a random oracle \( \mathcal{O}_H \) and a signing oracle \( \mathcal{O}_S \). Moreover, \( \mathcal{A} \) also can corrupt the designated verifier \( B \) and at most \( t - 1 \) signers \( A_1^*, \ldots, A_{t-1}^* \) in the signer group \( A \). In the end of the attack, the adversary \( \mathcal{A} \) outputs its forgery, then we checks whether it is a valid signature or not. We denote \( \text{Forge}_0 \) to be the event that the forged signature is valid and use the notation \( \text{Forge} \) for the same meaning in any game \( \text{Game}_i \). By definition, we have

\[
\epsilon = \text{Succ}^{\text{EF-CMA}}_{\text{TDS}, \mathcal{A}} = \Pr[\text{Forge}_0].
\]

**Game_1:** In this game, we guess \( t - 1 \) signers \( A_1^*, \ldots, A_{t-1}^* \) corrupted (controlled) by \( \mathcal{A} \) are \( A_1, \ldots, A_{t-1} \) from the signer group \( \mathcal{A} \). If the guess is right, we continue the game. Otherwise, we abort the game. Since the right guess probability is \( 1/\binom{n}{t-1} = \frac{(n-t)!}{(n-t+1)!} \), we will have

\[
\Pr[\text{Forge}_1] = \frac{(t-1)! \cdot (n-t+1)!}{n!} \Pr[\text{Forge}_0].
\]

**Game_2:** In this game, we simulate the KGen and KShare algorithms in the \( \mathcal{F} \mathcal{H} \mathcal{S} \) scheme. First, we choose a random number \( x_B \leftarrow \mathbb{Z}_q^* \) and replace the designated verifier \( B \)’s public key \( pk_B = Y_B = x_BP \) and private key \( sk_B = x_B \). Then, for the signer group \( \mathcal{A} \), we proceed the following steps.

1. We choose \( t - 1 \) random numbers \( x_1', x_2', \ldots, x_{t-1}' \leftarrow \mathbb{Z}_q^* \). Then for each signer \( A_i \), \( 1 \leq i \leq t - 1 \), we set \( pk_{A_i}' = Y_{A_i} = x_iP \) and \( sk_{A_i}' = x_i \). For the signer group \( \mathcal{A} \), we set the group public key \( pk_A \) as the challenge \( aP \), i.e. \( pk_A = Y_A = aP \).

2. We set a function \( L(x, y) \) as, for \( x, y \in \mathbb{N} \)

\[
L(x, y) = \begin{cases} 
\frac{x \cdot y}{\gcd(x, y)} \cdot \prod_{j=1}^{t-1} \frac{0 - j}{0 - j} \mod q, & 1 \leq x < t, \quad t \leq y \leq n \\
\prod_{j=1}^{t-1} \frac{0 - j}{0 - j} \mod q, & t \leq x = y \leq n 
\end{cases}
\]

Then, for other signers \( A_i \), \( t \leq i \leq n \), we compute

\[
\text{pk}_{A_i} = Y_{A_i} = \frac{1}{L(i, i)} \cdot \left( pk_A - \sum_{j=1}^{t-1} L(j, i) \text{pk}_{A_j}' \right)
\]
3. We choose another \( t - 1 \) random numbers \( x_1, x_2, \ldots, x_{t-1} \in \mathbb{Z}_q^* \). Then for each signer \( A_i \), \((1 \leq i \leq t - 1)\), we set \( pk_{A_i} = Y_{A_i} = x_iP \) and \( sk_{A_i} = x_i \).

4. We also choose random values \( pk_{A_1}, pk_{A_{t-1}} \in G_1 \) as the public keys of signers \( A_i, A_{1}, \ldots, A_{t-1} \). Then, we compute signer \( \tilde{A}_n \)'s public key \( pk_{\tilde{A}_n} = pk_{A_n} = \sum_{i=1}^{t-1} pk_{A_i} \).

5. In the end, we have the values \( pk_{A_i}, pk_{A_{1}}, \ldots, pk_{A_{t-1}}, 1 \leq i \leq n, \) and \( sk_{A_i}, sk_{A_{1}}, 1 \leq j \leq t - 1 \) and feed them and \((pk_{B}, sk_{B})\) to \( \mathcal{A} \).

With the above construction, the distributions of these values are unchanged. Even though \( \mathcal{A} \) corrupted the \( t - 1 \) signers \( A_1, A_2, \ldots, A_{t-1} \) and the designated verifier \( B \), he still can’t distinguish it from the real game. Therefore, we will have

\[
\Pr[\text{Forge}_2] = \Pr[\text{Forge}_1].
\]

**Game_5:** In this game, we will simulate the random oracle \( \mathcal{O}_H \) by maintaining a hash list \( A_H \). In order to embed the challenge \( bP \in G_1 \) into oracle answer, when a fresh query \((R,m)\) is queried, we choose a random number \( h \leftarrow \mathbb{Z}_q^* \) and compute \( H(R||m) = hbP \). We then store \((R,m,h,hbP)\) in the hash list \( A_H \) and return \( H(R||m) \) as the answer to the oracle query. Clearly, in the random oracle model, \( h \) is randomly chosen from \( \mathbb{Z}_q^* \), then \( H(R||m) \) is uniformly distributed in \( G_1 \), and this game is therefore perfectly indistinguishable from the previous one. Hence,

\[
\Pr[\text{Forge}_3] = \Pr[\text{Forge}_2].
\]

**Game_4:** In this game, we only keep executions which outputs a valid signature/message \((\sigma = (R,S),m)\) such that \((R,m)\) has been queried from \( \mathcal{O}_H \). This will make a difference only if \( \sigma = (R,S) \) is a valid signature on \( m \), while \((R,m)\) has not been queried from \( \mathcal{O}_H \). Since \( H(R||m) \) is uniformly distributed in \( G_1 \), the verification equation

\[
e\left( S + \frac{1}{x_B} R, P \right) = e(H(R||m), Y_A)
\]

happens with probability \( 1/q \). Therefore, we will have

\[
\left| \Pr[\text{Forge}_4] - \Pr[\text{Forge}_3] \right| \leq \frac{1}{q}.
\]

**Game_5:** In this game, we simulate total \( q \), times signing oracle \( \mathcal{O}_S \). For any fresh \( m \), whose signature is queried, we choose a random number \( r \leftarrow \mathbb{Z}_q^* \) and compute \( R = rY_B \). If the hash list \( A_H \) includes a 4-tuple \((R,m,h,hP)\), we abort the simulation, otherwise, we choose another random number \( h \leftarrow \mathbb{Z}_q^* \), set \( H(R||m) = hbP \) and store the 4-tuple \((R,m,h, hbP)\) in the hash list \( A_H \). We also compute \( S = hY_A - rP = haP - rP \), then \( \sigma = (R,S) \) provides a valid signature of \( m \). In this game, if it does not abort, this new oracle perfectly simulates the signature. As we abort with probability at most \( q_{\text{ch}}/q \), we will have

\[
\left| \Pr[\text{Forge}_5] - \Pr[\text{Forge}_4] \right| \leq \frac{q_{\text{ch}} \cdot q}{q}.
\]

At the end of Game_5, we have completed the simulation of random oracle \( \mathcal{O}_H \) as well as the signing oracle \( \mathcal{O}_S \). When the adversary \( \mathcal{A} \) outputs a forgery \((\sigma = (R^*, S^*), m^*)\), we look up the hash list \( A_H \) and find out the entry \((R,m,h, hbP)\) such that \( R = R^*, m = m^* \). Then the output of the CDH problem challenge is

\[
abP = \frac{1}{h} \left( S^* + \frac{1}{x_B} R^* \right)
\]

We then conclude that

\[
\Pr[\text{Forge}_5] \leq \text{Adv}^{\text{CDH}}_{G_1}(\mathcal{A}).
\]

From Eqs. (12)–(18), we have

\[
e' \geq \frac{(t-1)! (n-t+1)!}{n!} \cdot e - \frac{1}{q} - \frac{q_{\text{ch}} \cdot q}{q}.
\]
We only consider the time-consuming scale multiplication in $G_1$ in the simulation. Then, there are $n + t - r^2 + 3t - 1$ scale multiplications in Game$_2$, $q_b$ scale multiplications in Game$_3$ and scale multiplication $4q_b + 2$ in Game$_5$. By a simple computation, we can obtain the claimed bound for
\[ t' \leq t + (nt - r^2 + 3t + 1 + q_b + 4q_b)T_{\text{smul}}. \]
Thus, the proof is completed. \( \square \)

Theorem 3. The proposed \( TDF \) scheme is indeed a directed signature scheme.

Proof. Here, we informally prove that the \( TDF \) scheme is indeed a directed signature scheme. First, we consider the Invisibility by the game defined in Section 2.3. Let \( D \) be a distinguisher and \( CH \) be a challenger. In the find stage, \( D \) chooses two messages \( m_0 \) and \( m_1 \), and sends them to \( CH \). \( CH \) then chooses \( b \in \{0, 1\} \) and invokes the indeed the signer group \( A \) to make a valid directed signature
\[ \sigma = (R = rY_b, S = f'(0)H(R||m_b) - rP) \]
on \( m_b \). Finally, \( CH \) returns \( \sigma = (R, S) \) to \( D \). In the guess stage, \( D \) returns his guess \( b' \in \{0, 1\} \) to \( CH \). Observe the verification equation
\[ e\left(S + \frac{1}{x_b}R, P\right) = e(H(R||m_b), Y_A) \]
If \( D \) have non-negligible advantage \( \text{Adv}_{TDF,D} = 2\text{Pr}[b' = b] - 1 \) to guess the right \( b \), then he must know the value of \( \frac{1}{x_b}R = rP \). However, from \( R = rY_b = rx_bP \) and \( Y_B = x_bP \), to compute \( rP \) is an ICDH problem in \( G_1 \). Therefore, due to the hardness of ICDH problem, \( D \) cannot guess the right \( m_b \) from the verification equation. Thus the property of Invisibility is satisfied.

On the other hand, the property of Transitivity also clearly follows. Since both the signer group \( A \) and the designated verifier \( B \) can compute \( rP \). Then anyone \( C \), with the help of \( A \) or \( B \), can verify the signature \( \sigma \) from the verification equation. Therefore, our proposed \( TDF \) scheme is indeed a directed signature scheme. This completes the proof. \( \square \)

6. Performance discussion

Performance of signature protocols can be approximated in terms of computation and communication overheads. In this section, we mainly discuss the performance of our proposed \( TDF \) scheme.

For convenience, the following notations are used to analyze the computation and communication complexity. \( T_{\text{smul}} \) represents the time for one scale multiplication computation in \( G_1 \). \( T_{\text{pair}} \) denotes the time for one pairing computation; \( T_{\text{mhash}} \) define the time for one Map-to-Point hash function; \( N_t \) denotes the total number of broadcasts and \( N_b \) denotes the total number of broadcasts. Note that the times for other computation operations are ignored, since they are much smaller than \( T_{\text{smul}} \), \( T_{\text{pair}} \) and \( T_{\text{mhash}} \).

We summarize the computation and communication overheads of our proposed \( TDF \) scheme Table 1. As shown in Table 1, the computation complexity for KGen, KShare, TSign, DVerify and PVerify are \((n + 1)T_{\text{smul}}, 2nt T_{\text{smul}}, (2t + 1)T_{\text{smul}} + T_{\text{mhash}} + (2t - 2)T_{\text{pair}}, T_{\text{smul}} + T_{\text{mhash}} + 2T_{\text{pair}}, \) and \( T_{\text{mhash}} + 4T_{\text{pair}} \) in our \( TDF \) scheme, respectively. Also the total communication overheads are \( 2nN_b + (n^2 - n)N_t \) and \( N_b + (t - 1)N_t \) for KShare and TSign in our \( TDF \) scheme, respectively.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Computation overheads</th>
<th>Communication overheads</th>
</tr>
</thead>
<tbody>
<tr>
<td>KGen</td>
<td>((n + 1)T_{\text{smul}})</td>
<td>(2nN_b + (n^2 - n)N_t)</td>
</tr>
<tr>
<td>KShare</td>
<td>(2nt T_{\text{smul}})</td>
<td>(N_b + (t - 1)N_t)</td>
</tr>
<tr>
<td>TSign</td>
<td>((2t + 1)T_{\text{smul}} + T_{\text{mhash}} + (2t - 2)T_{\text{pair}})</td>
<td></td>
</tr>
<tr>
<td>DVerify</td>
<td>(T_{\text{smul}} + T_{\text{mhash}} + 2T_{\text{pair}})</td>
<td></td>
</tr>
<tr>
<td>PVerify</td>
<td>(T_{\text{mhash}} + 4T_{\text{pair}})</td>
<td></td>
</tr>
</tbody>
</table>
7. Conclusions

The directed signature, due to its merits, can replace standard signatures and prevent potential misuse of signatures. In this paper, we have formally defined the threshold directed signature, then proposed a new \((t,n)\) threshold directed signature \(TDS\) scheme based on the bilinear pairing, and use the techniques from provable security to analyze the security of our proposed scheme. Due to its merits, the proposal is suitable for some applications where messages signed are personally or commercially sensitive to the signature receiver.

Acknowledgement

The author would like to thank the anonymous referees for their valuable comments and suggestions to improve the presentation of this paper.

References