Practical Threshold Signatures without Random Oracles

Jin Li[1] ⋆, Tsz Hon Yuen[2] and Kwangjo Kim[1]

[1] International Research center for Information Security (IRIS)
Information and Communications University(ICU)
58-4 Hwaam-dong Yusong-ku, Taejon, 305-732, Korea
[2] School of Information Technology and Computer Science
University of Wollongong, NSW 2522 Australia

Abstract. We propose a secure threshold signature scheme without trusted dealer. Our construction is based on the recently proposed signature scheme of Waters in EUROCRYPT'05. The new threshold signature scheme is more efficient than the previous threshold signature schemes without random oracles. Meanwhile, the signature share generation and verification algorithms are non-interactive. Furthermore, it is the first threshold signature scheme based on the computational Diffie-Hellman (CDH) problem without random oracles.

Keywords: Threshold Signature, Bilinear groups, CDH problem

1 Introduction

Digital signatures can be produced by a group of players rather then by one party by using a threshold signature scheme. In contrast to the regular signature schemes where the signer is a single entity which holds the secret key, in (k, n)-threshold signature schemes the secret key is shared by a group of k players. In order to produce a valid signature on a given message m, individual players produce their partial signatures on that message, and then combine them into a full signature on m. A distributed signature scheme achieves threshold k, if no coalition of $k-1$ (or less) players can produce a new valid signature, even after the system has produced many signatures on different messages. A signature resulting from a threshold signature scheme is the same as if it was produced by a single signer possessing the full secret signature key. In particular, the validity of this signature can be verified by anyone who has the corresponding unique public verification key. In other words, the fact that the signature was produced in a distributed fashion is transparent to the recipient of the signature.

Threshold cryptography and secret sharing have been given considerable attention since they were proposed. The first threshold secret sharing schemes,

⋆ This work was partially supported by the 2nd stage of Brain Korea 21 Project sponsored by the Ministry of Education and Human Resources Development, Korea

based on the Lagrange interpolating polynomial and linear project geometry, were proposed by Shamir [11]. Many efficient digital signature and threshold signature schemes are proved secure in the random oracle model. However, several papers proved that some popular cryptosystems previously proved secure in the random oracle are actually provably insecure when the random oracle is instantiated by any real-world hashing functions [2]. Therefore, provably secure threshold signature scheme in the standard model attracts a great interest.

Related Work. Recently, [13] gave the first threshold signature without random oracles. However, the threshold signature scheme requires that the users generate the signature interactively. Meanwhile, the correctness of these generated signature shares cannot be verified. Ideally, there is no other interaction in the threshold signature scheme, namely the players need not talk to each other during signing. Such threshold systems are called non-interactive. Often one requires that threshold signature be robust [8], namely if threshold signature fails, the combiner can identify the signing players that supplied invalid partial signatures. In [12], a practical threshold signature scheme based on RSA was proposed, which is non-interactive. However, it required a trusted dealer.
Contributions. In this paper, we propose a new practical threshold signature scheme without trusted dealer. The threshold signature has the following properties:

1. It is provably secure without relying on the random oracle model;
2. Signature share generation and verification are completely non-interactive;
3. The scheme is the first threshold signature scheme based on the CDH problem without random oracles;
4. Signature share generation and verification algorithms are very efficient.

2 Preliminaries

2.1 Security Definitions and Notions

We shows the definition as follows:

Definition 1. *A (k, n)-threshold signature scheme consists of algorithms (DKG, SS, SV, SC, Vrfy). These algorithms are specified as follows:*

1. *DKG is the distributed key generation algorithm. On input security parameter 1^λ, k, n it outputs public key pk and secret key sk. Meanwhile, it also outputs the private value sk_i and verification key vk_i of player i such that the values $(sk_1, \cdots, sk_n)$ form a (k, n)-threshold secret sharing of sk. The public output of the protocol contains the public key pk and verification key $VK = (vk_1, \cdots, vk_n)$.*
2. *SS is the signature share generation algorithm run by player i, on input secret share sk_i, a message m, it returns σ_i as the shared signature.*
3. *SV is the signature share verification, on input public key pk, verification key vk_i, a message m, σ_i, output 1 if it is valid. Otherwise, output 0.*

4. *SC is the signature share combining algorithm, on input $|\Phi|$ different shares $\{\sigma_i\}_{i\in\Phi}$, where $\Phi \subset \{1, 2, \cdots, n\}$ is a set and $|\Phi| \geq k$, a message m, it returns σ as the signature.*
5. *Vrfy is the signature verification algorithm, on input pk, m, σ, returns 1 if it is valid, otherwise, returns 0.*

DKG makes use of an appropriate distributed secret-sharing technique to generate shares of the private key as well as verification keys that will be used for checking the validity of signature shares. The signing server then keeps their private key shares secret but publishes the verification keys. Given a message for signing, the signing servers then run the signature share generation algorithm SS taking the message as input and send the resulting signature shares to the combiner. Note that the validity of the shares can be checked by running the signature share verification algorithm SV. When the user collects valid signature shares from at least k servers, the signature can be reconstructed by running the share combining algorithm SC. Notice that our model explicitly requires that the generation and verification of signature shares is completely non-interactive.

We work with a static corruption model: the adversary must choose which players to corrupt at the very beginning of the attack.

Unforgeability for (k, n)-threshold signature is defined as in the following game involving an adversary $\mathcal{A}$.

We have a set of n players, indexed $1, \cdots, n$, a trusted dealer, and an adversary $\mathcal{A}$. There is also a share signing algorithm SS, a share verification algorithm SV, a share combining algorithm SC, and a signature verification algorithm Vrfy.

At the beginning of the game, the adversary selects a subset of $k-1$ players to corrupt. In the dealing phase, the dealer generates a public key pk along with secret key shares $sk_1, \cdots, sk_n$, and verification keys $\mathsf{VK} = \{vk_1, \cdots, vk_n\}$. The adversary obtains the secret key shares of the corrupted players, along with the public key and verification keys. After the dealing phase, the adversary submits signing requests to the uncorrupted players for messages of his choice. Upon such a request, a player outputs a signature share for the given message.

We say that the adversary forges a signature if at the end of the game he outputs a valid signature on a message that was not submitted as a signing request to the uncorrupted players. We say that the threshold signature scheme is unforgeable if it is computationally infeasible for the adversary to forge a signature.

2.2 Pairings and Problem

Let $\mathbb{G}, \mathbb{G}_T$ be cyclic groups of prime order p, writing the group action multiplicatively. Let g be a generator of $\mathbb{G}$. A bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is also defined.

Definition 2. *(Computational Diffie-Hellman* **CDH** *Assumption) The Computational Diffie-Hellman problem is that, given g, $g^x, g^y \in (\mathbb{G})^3$ for unknown $x, y \in \mathbb{Z}_p^*$, it is hard to compute g^{xy}.*

2.3 Brief Review of Waters Signature Scheme

In EUROCRYPT'05, Waters [14] proposed an identity based encryption scheme. From the private key extraction algorithm, a signature scheme without random oracles has been constructed [14].

1. **Gen**. Choose $\alpha \in Z_p$ and let $g_1 = g^\alpha$. Additionally, two random values $g_2, u' \in \mathbb{G}$ and a random n-length vector $\mathsf{U} = (u_i)$, whose elements are chosen at random from $\mathbb{G}$. The public key is $pk = (g_1, g_2, u', \mathsf{U})$ and the secret key is g_2^α.
2. **Sign**. To generate a signature on message $M = (\mu_1, \cdots, \mu_n) \in \{0,1\}^n$, pick $s \in_R Z_p^*$ and output the signature as $\sigma = (g_2^\alpha \cdot (u' \prod_{j=1}^n u_j^{\mu_j})^s,\ g^s)$ with his secret key g_2^α.
3. **Verify**. Given a signature σ on message $M = (\mu_1, \cdots, \mu_n) \in \{0,1\}^n$, it first parses $\sigma = (\sigma_1, \sigma_2)$. Then it checks if the following equation holds: $\hat{e}(\sigma_1, g) = \hat{e}(g_2,\ g_1) \cdot \hat{e}\ (u' \prod_{i=1}^n u_i^{\mu_i},\ \sigma_2)$. Output 1 if it is valid. Otherwise, output 0.

2.4 Brief Review of GJKR's DKG

Before we give the description of GJKR's DKG, we review two fundamental secret sharing schemes:

A. **Shamir's Secret Sharing [11]**: Given a secret α, choose at random a degree $k-1$ polynomial function $f \in Z_p[X]$ such that $x = f(0)$. Give to player P_i a share $x_i = f(i) \bmod p$, where p is a prime. We will write $(x_1, \cdots, x_n) \leftrightarrow (x)$ to denote such a sharing.

B. **Feldman Verifiable Secret Sharing [6]**: Like Shamir's secret sharing scheme, it generates for each player P_i a share $x_i = f(i) \bmod p$, such that $(x_1, \cdots, x_n) \leftrightarrow (x)$. If $f(x) = \sum_{i=0}^{k-1} a_i x^i$, then the dealer broadcasts the values $A_i = g^{a_i}$, where g is subgroup generator. This will allow the players to check that the values x_i really define a secret by checking that $g^{x_i} = \prod_{j=0}^{k-1} A_j^{i^j}$. It will also allow detection of incorrect shares at reconstruction time. In the following we will refer to this protocol by Feldman-VSS.

Pedersen proposed a DKG protocol in [9]. The basic idea in Pedersen's DKG protocol is to have n parallel executions of Feldman-VSS protocol in which each player P_i acts as a dealer of a random secret z_i that he picks. The secret value x is taken to be the sum of the properly shared $z_i's$. Since Feldman-VSS has the additional property of revealing $y_i = g^{z_i}$, the public value y is the product of the y_i's that correspond to those properly shared $z_i's$.

In spite of its use in many protocols, Pedersen's DKG [9] cannot guarantee the correctness of the output distribution in the presence of an adversary. Specifically, Gennaro et al. [7] showed a strategy for an adversary to manipulate the distribution of the resulting secret x to something quite different from the uniform distribution. In contrast to the Pedersen's DKG, Gennaro et al. [7] presented the GJKR's DKG protocol that enjoys a full proof of security. It starts by

running a commitment stage where each player P_i commits to a $(k-1)$-degree polynomial $f_i(z)$ whose constant coefficient is the random value, z_i, contributed by P_i to the jointly generated secret α. To realize the above commitment stage it used the information-theoretic verifiable secret sharing protocol due to Pedersen's DKG. After the value x is fixed the parties can efficiently and securely compute $y = g^x$. Most importantly, this guarantees that no bias in the output x or y of the protocol is possible, and it allows to present a full proof of security based on a careful simulation argument. Each honest party P_j computes its share x_j of x, and we have that for the set of shares R: $x = \sum_{j \in R} \lambda_j x_j$. Meanwhile, for each share x_j, the value g^{x_j} can be computed from publicly available information broadcast.

We now describe in detail the secure distributed key generation [7](GJKR's DKG):

1. In order to generating a secret key x, each player P_i performs interactively as follows:
   (a) P_i chooses two random polynomials $f_i(z)$, $f'_i(z)$ over Z_p of degree $k-1$: $f_i(z) = a_{i0} + a_{i1}z + \cdots + a_{i,k-1}z^{k-1}$, $f'_i(z) = b_{i0} + b_{i1}z + \cdots + b_{i,k-1}z^{k-1}$. Let $z_i = a_{i0} = f_i(0)$. P_i broadcasts $C_{it} = g^{a_{it}} h^{b_{ik}} \bmod p$ for $t = 0, \cdots, k-1$. P_i computes the shares $s_{ij} = f_i(j)$, $s'_{ij} = f'_i(j) \bmod p$ for $j = 1, \cdots, n$ and sends s_{ij}, s'_{ij} to player P_j.
   (b) Each player P_j verifies the shares he received from the other players. For each $i = 1, \cdots, n$, P_j checks if $g^{s_{ij}} h^{s'_{ij}} = \prod_{t=0}^{k-1} (C_{it})^{j^t} \bmod p$. If the check fails for an index i, P_j broadcasts a complaint against P_i.
   (c) Each player P_i who, as a dealer, received a complaint from player P_j broadcasts the values s_{ij}, s'_{ij}.
   (d) Each player marks as disqualified any player that either received more than $k-1$ complaints in Step 1b, or answered to a complaint in Step 1c with invalid values.
   (e) Each player P_i then builds the same set of non-disqualified players $QUAL$ and sets his share of the secret as $x_i = \sum_{i \in QUAL} s_{ji} \ mod \ p$, and the value $x'_i = \sum_{i \in QUAL} s'_{ji} \bmod p$.
2. Finally, they extract $y = g^x \bmod p$ as follows:
   (a) Each player $i \in QUAL$ exposes $y_i = g^{z_i} \bmod p$ via Feldman VSS and broadcasts $A_{it} = g^{a_{it}} \bmod p$ for $t = 0, \cdots, k-1$. Then P_j verifies the values broadcast by the other players in $QUAL$, namely, for each $i \in QUAL$, P_j checks if $g^{s_{ij}} = \prod_{t=0}^{k-1} (A_{it})^{j^t} \bmod p$. If the check fails for an index i, P_j complains against P_i by broadcasting the values s_{ij}, s'_{ij}.
   (b) For players P_i who receive at least one valid complaint, the other players run the reconstruction phase of Pedersen-VSS to compute $z_i, f_i(z), A_{it}$ for $t = 0, \cdots, k-1$ in the clear.
   (c) For all players in $QUAL$, set $y_i = A_{i0} = g^{z_i} \bmod p$. Compute $y = \prod_{i \in QUAL} y_i \bmod p$.
   The above argument shows that the secret x can be efficiently reconstructed, via interpolation, out of any k correct shares.

We need to show that we can tell apart correct shares from incorrect ones. For this we show that for each share x_j, the value g^{x_j} can be computed from publicly available information broadcast in Step 2*a*: $g^{x_j} = g^{\sum_{i\in QUAL} s_{ij}} = \prod_{i\in QUAL} g^{s_{ij}}$ $= \prod_{i\in QUAL} \prod_{t=0}^{k-1} (A_{it})^{jt} \bmod p$. Thus the publicly available value g^{x_j} makes it possible to verify the correctness of share x_j at reconstruction time.

Meanwhile, for any set R of k correct shares, $z_i = \sum_{j\in R} \lambda_j \cdot s_{ij} \bmod p$, where λ_j are appropriate Lagrange interpolation coefficients for the set R. Since each honest party P_j computes its share x_j as $x_j = \sum_{i\in QUAL} s_{ij}$, then we have that for the set of shares R: $x = \sum_{i\in QUAL} z_i = \sum_{i\in QUAL} (\sum_{j\in R} \lambda_j \cdot s_{ij}) = \sum_{j\in R} \lambda_j \cdot (\sum_{i\in QUAL} s_{ij}) = \sum_{j\in R} \lambda_j x_j$.

3 The Threshold Signature Scheme With Trusted Dealer

Let $\mathbb{G}$ be a bilinear group of prime order p. Given a pairing: $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. A random generator $g \in \mathbb{G}$ is also selected.

1. **DKG.**To generate public key, the trusted dealer picks $\alpha \in Z_p$ and computes $g_1 = g^\alpha$. Additionally, two random values $g_2, u' \in \mathbb{G}$ and a random n-length vector $\mathsf{U} = (u_i)$, whose elements are chosen at random from $\mathbb{G}$, are also generated.
   a. It chooses a $k-1$ degree function $f(x) \in Z_p(x)$ such that $\alpha = f(0)$ and computes n secret key share (i, sk_i) for $1 \le i \le n$ by using Shamir secret sharing scheme, which is defined as $sk_i = g_2^{f(i)}$.
   b. The public verification key VK consists of the n-tuple $(g^{f(1)}, \cdots, g^{f(n)})$. Then, it sends to player P_i a share $g_2^{f(i)}$ for $1 \le i \le n$.
   c. The public key is $(g_1, g_2, u', \mathsf{U}, \mathsf{VK})$ and the secret key shares are sk_i for $1 \le i \le n$.
2. **SS**. To generate a signature on message $M = (\mu_1, \cdots, \mu_n) \in \{0,1\}^n$, player i picks $r_i \in_R Z_p^*$ and outputs the partial signature as $\sigma_i = (sk_i \cdot (u' \prod_{j=1}^n u_j^{\mu_j})^{r_i}$, $g^{r_i})$ with its secret key share sk_i.
3. **SV**. On input $\sigma_i = (\sigma_{i,1}, \sigma_{i,2})$, verification key vk_i, the verifier checks if the following equation holds: $\hat{e}(\sigma_{i,1}, g) = \hat{e}(g_2, vk_i) \cdot \hat{e}\ (u' \prod_{j=1}^n u_j^{\mu_j}, \sigma_{i,2})$. Output 1 if it is valid. Otherwise, output 0.
4. **SC**. Let $\lambda_1, \cdots, \lambda_k \in \mathbb{Z}_p$ be the Lagrange coefficients so that $\alpha = f(0) = \sum_{i=1}^k \lambda_i f(i)$. Assume signature share combination algorithm has $|\Phi|$ valid signature shares $\sigma_i = (\sigma_{i,1}, \sigma_{i,2})$, where $|\Phi| \ge k$. Without loss of generality we assume that player $i = 1, \cdots, k$ were used to generate the shares. The signature combination algorithm computes the signature on message M as $\sigma = (\prod_{i=1}^k (\sigma_{i,1})^{\lambda_i}, \prod_{i=1}^k (\sigma_{i,2})^{\lambda_i})$.
5. **Vrfy**. Given a signature σ on message $M = (\mu_1, \cdots, \mu_n) \in \{0,1\}^n$, it first parses $\sigma = (\overline{\sigma}_1, \overline{\sigma}_2)$. Then it checks if the following equation holds: $\hat{e}(\overline{\sigma}_1, g) = \hat{e}(g_2,\ g_1) \cdot \hat{e}\ (u' \prod_{i=1}^n u_i^{\mu_i},\ \overline{\sigma}_2)$. Output 1 if it is valid. Otherwise, output 0.

3.1 Efficiency Analysis

The new threshold signature scheme is non-interactive. Furthermore, signature share generation algorithm requires only two exponentiation computation for each player. Though [12] also gave a practical non-interactive threshold signature scheme with trusted dealer based on RSA problem, it required one exponentiation with zero-knowledge proof, which is actually not very efficient. Recently, a short threshold signature scheme [13] has been proposed, however, it is very inefficient for it requires the players generate signature shares interactively.

3.2 Security Result

Theorem 1. *Under the CDH assumption, the proposed practical threshold signature scheme is a secure (unforgeable and robust) threshold signature scheme resistant to $k-1$ faults against a static malicious adversary, when the number of player is $n \geq 2k-1$.*

Proof. Our algorithm $\mathcal{C}$ described below solves CDH problem for a randomly given instance $\{g, X = g^x, Y = g^y\}$ and asked to compute g^{xy}.
Setup: First, $\mathcal{C}$ defines $g_1 = X$ and sets an integer, $m = 4q_S$, chooses an integer, k', uniformly at random between 0 and n. Choose a random n-length vector, $\overrightarrow{a} = (a_i)$, all are chosen uniformly at random between 0 and $m-1$. Then, the simulator chooses a random $b' \in Z_p$ and an n-length vector, $\overrightarrow{b} = (b_i)$, where the elements of $\overrightarrow{b}$ are chosen at random in Z_p. It then assigns $u' = g_1^{p-km+a'} g^{b'}$ and the parameter U as $u_i = g_1^{a_i} g^{b_i}$. The system parameters $\mathsf{params} = (g, g_1, u', (u_i))$ are sent to $\mathcal{A}$. Two pairs of functions are defined for a message $M = \{\mu_1, \cdots, \mu_n\} \in \{0,1\}^n$. We define $F(M) = (p - mk) + a' + \sum_{i=1}^{n} a_i^{\mu_i}$. Next, we define $J(M) = b' + \sum_{i=1}^{n} b_i^{\mu_i}$. Finally, define a binary function K(M) as $K(M) = \begin{cases} 0, \text{ if } a' + \sum_{i=1}^{n} a_i^{\mu_i} \equiv 0 \ (mod\ m); \\ 1, \text{ otherwise.} \end{cases}$

We assume w.l.o.g. that the adversary corrupted the first $k-1$ players $P_1, \cdots, P_{k-1}$. Then, C generates the secret key shares for the $k-1$ corrupt players in S. To do so, C first picks $k-1$ random integers $x_1, \cdots, x_{k-1} \in Z_p$. Let $f \in Z_p[X]$ be the degree $k-1$ polynomial implicitly defined to satisfy $f(0) = x$ and $f(i) = x_i$ for $i = 1, \cdots, k-1$. Algorithm C gives A the $k-1$ secret key shares $sk_i = g_2^{x_i}$. These keys are consistent with this polynomial f since $sk_i = g_2^{f(i)}$ for $i = 1, \cdots, k-1$.

Finally, C constructs the verification key VK, which is a n-vector $(vk_1, \cdots, vk_n)$ such that $vk_i = g^{f(i)}$ for the polynomial f defined above, as follows:

For $i \in S$, computing vk_i is easy since $f(i)$ is equal to one of the $x_1, \cdots, x_{k-1}$, which are known to C. Thus, $vk_1, \cdots, vk_{k-1}$ are easy for C to compute.

For $i \notin S$, algorithm C needs to compute the Lagrange coefficients $\lambda_{0,i}, \lambda_{1,i}$, $\cdots$, $\lambda_{k-1,i} \in Z_p$ such that $f(i) = \lambda_{0,i} f(0) + \sum_{j=1}^{k-1} \lambda_{j,i} f(j)$; these Lagrange coefficients are easily calculated since they do not depend on f. Algorithm C then sets $vk_i = g_1^{\lambda_{0,i}} vk_1^{\lambda_{1,i}} \cdots vk_{k-1}^{\lambda_{k-1,i}}$, which entails that $vk_i = g^{f(i)}$ as required.

Once it has computed all the vk_i's, C gives to A the verification key $\mathsf{VK} = (vk_1, \cdots, vk_n)$.

Signature Share Query: A issues up to q_S signature share generation queries to the uncorrupt players. Consider a signature share generation query to player $i \notin S$. Let $M=(\mu_1, \cdots, \mu_n) \in \{0,1\}^n$ be the message for signature share query. If $K(M) = 0$, $\mathcal{C}$ will abort. Otherwise, $\mathcal{C}$ computes the simulated signature share for M as follows: Algorithm B needs to return $(i, (\sigma_{i,0}, \sigma_{i,1}))$ where $\sigma_{i,0} = g_2^{x_i} \cdot (u' \prod_{j=1}^{n} u_j^{\mu_j})^{r_i}$, $\sigma_{i,1}=g^{r_i}$.

To do so, B first computes the Lagrange coefficients $\lambda_0, \lambda_1, \cdots, \lambda_{k-1} \in Z_p$ such that $f(i) = \lambda_{0,i} f(0) + \sum_{j=1}^{k-1} \lambda_{j,i} f(j)$. Pick $r_i' \in \mathbb{Z}_p^*$ and output the simulated signature share as $\sigma_i = (g_2^{-\lambda_{0,i}\frac{J(M)}{F(M)}} (u' \prod_{i=1}^{n} u_i)^{r_i'} \cdot g_2^{\sum_{j=1}^{k-1} \lambda_{j,i} f(j)}, g_2^{\frac{-\lambda_{0,i}}{F(M)}} g^{r_i'})$. The correctness of the signature can be easily verified.

Finally, the adversary outputs a forged signature (σ_1^*, σ_2^*) on message $M^* = (\mu_1^*, \cdots, \mu_n^*)$. If $a' + \sum_{i=1}^{n} a_i^{\mu_i^*} \neq km$, the challenger will abort. Otherwise, $\mathcal{C}$ will compute $g^{xy} = \frac{\sigma_1^*}{(\sigma_2^*)^{J(M)}}$.

For the simulation to complete without aborting, we require that all signature queries on M will have $K(M) \neq km$, that forgery signature on message M^* has $K(M^*) = 0 \ mod \ p$. In fact, the probability analysis is very similar to [23]. So, we can get the probability of solving computational CDH problem as $\epsilon' = \frac{\epsilon}{16(q_E+q_S)q_S(n+1)(m+1)}$ if the adversary success with probability ϵ.

4 The Threshold Signature Scheme Without Trusted Dealer

We have construct a threshold signature scheme with trusted dealer in last section. However, in some situations, it does not have trusted dealer. So, in order to generate threshold signature, the players should generate the public key jointly. We assume that the involved n participants are connected by a broadcast channel. Furthermore, any one pair of the participants is connected by a private channel. We also assume that there is a universal clock such that each participant knows the absolute time, and the communication channel is (partially) synchronous by rounds.

It is also assumed that an adversary can corrupt up to $k-1$ of the n players in the network, for any value of $k-1 < \frac{n}{2}$ (this is the best achievable threshold or resilience for solutions that provide both secrecy and robustness). We consider a malicious adversary that may cause corrupted players to divert from the specified protocol in any way. We assume that the computational power of the adversary is adequately modelled by a probabilistic polynomial time Turing machine. Furthermore, we consider a static adversary who chooses corrupted participants at the beginning of each time period. For the robustness, it means that the scheme can be successfully finished even if the adversary corrupts $k-1$ participants at most.

GJKR's DKG protocol of [7] is based on the ideas similar to the protocol of Pedersen [9], has comparable complexity, but provably fixes the weakness of the

latter. So, we use the GJKR's DKG protocol in [7] to distributedly generate the shared secret keys and output public keys. The system parameters are the same with the scheme in section 3.

- **DKG**. To generate public key, n servers jointly generate user public key $g_1 = g^{\alpha}$ by using GJKR's DKG. Meanwhile, Each player P_i broadcasts $g^{f(i)}$ for a random jointly generated degree $k-1$ polynomial $f \in Z_p[X]$ such that $\alpha = f(0)$. Additionally, two values $g_2, u' \in \mathbb{G}$ and a n-length vector $\mathsf{U} = (u_i)$, whose elements are from $\mathbb{G}$, are also generated by using GJKR's DKG algorithm, respectively. Furthermore, player P_i gets its secret share $sk_i = g_2^{f(i)}$ for $1 \leq i \leq n$. The public verification key VK=$(vk_1, \cdots, vk_n)$ consists of the n-tuple $(g^{f(1)}, \cdots, g^{f(n)})$. The public key is $(g_1, g_2, u', \mathsf{U}, \mathsf{VK})$ and the shared secret keys are sk_i for $1 \leq i \leq n$.
- **SS**, **SV**, **SC**, **Vrfy** algorithms are the same with section 2.4.

Correctness is obvious. Next, we will prove its robustness and unforgeablity.

4.1 Security Result

We also prove the unforgeability by using the concept of simulatable adversary view [16] proposed by Gennaro et al.

Theorem 2. *Under the CDH assumption, the proposed practical threshold signature scheme is a secure (unforgeable and robust) threshold signature scheme resistant to $k-1$ faults against a static malicious adversary, when the number of player is $n \geq 2k-1$.*

Proof. The robustness is evident.

The construction of DKG is the same with [7], which has been proved to be simulatable. Next, we prove the protocol SS is simulatable:

Given public key $(g_1, g_2, u', \mathsf{U}, \mathsf{VK})$, message $m = (\mu_1, \cdots, \mu_n) \in \{0,1\}^n$, signature $\sigma = (\overline{\sigma}_1, \overline{\sigma}_2)$, $k-1$ shares $(\alpha_1, \cdots, \alpha_{k-1})$ of the corrupted players, it picks random values $r_i \in Z_p$ and computes $\sigma_i = g_2^{\alpha_i} \cdot (u' \prod_{j=1}^{n} u_j^{\mu_j})^{r_i}$, $g^{r_i})$ for $i = 1, \cdots, k-1$. From the values $\sigma = (\overline{\sigma}_1, \overline{\sigma}_2)$, and σ_i for $i = 1, \cdots, k-1$, simulator generates $\sigma_j = \frac{\sigma}{\sigma_i^{\lambda_{j,i}}}$, for $j = k, \cdots, n$, with known Lagrange interpolation coefficients $\lambda_{j,i}$.

5 Conclusion

A secure threshold signature scheme without trusted dealer is proposed in this paper. Our construction is based on the recently proposed signature scheme of Waters [14], combined with the new technique [3]. It is provably secure without relying on the random oracle model. Additionally, signature share generation and verification is completely non-interactive. The new threshold signature scheme is more efficient than the previous threshold signature schemes without random oracles. Furthermore, it is the first threshold signature scheme based on the CDH problem without relying on random oracles.

References

1. M. Abe and S. Fehr. Adaptively secure Feldman VSS and applications to universally-composable threshold cryptography. In Advances in Cryptology-CRYPTO 2004, LNCS 3152, Springer-Verlag, pp. 317-334 ,2004.
2. M. Bellare, A. Boldyreva, and A. Palacio. *An Uninstantiable Random-Oracle-Model Scheme for a Hybrid-Encryption Problem.* In Advances in Cryptology-EUROCRYPT 2004, LNCS 3027, pages 171-188. Springer, 2004.
3. D. Boneh, X. Boyen and S. Halevi. Chosen ciphertext secure public key threshold encryption without random oracles. CT-RSA'05. LNCS 3860, pp. 226-243, springer, 2006.
4. R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Adaptive security for threshold cryptosystems. In Advances in Cryptology-CRYPTO'99, LNCS 1666, Springer-Verlag, pp. 98-115, 1999.
5. Y. Desmedt and Y. Frankel. Threshold cryptosystems. In Advances in Cryptology-Crypto'89, pages 307-315. LNCS 435, Springer-Verlag, 1989.
6. P. Feldman. A Practical Scheme for Non-Interactive Verifiable Secret Sharing. In Proc. 28th FOCS, pages 427-437.
7. R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, Secure Distributed Key Generation for Discrete-Log Based Cryptosystem, In Advances in Cryptology-EUROCRYPT'99, LNCS 1592, Springer-Verlag, pages 295-310, 1999.
8. R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin. Robust threshold DSS signatures. Information and Computation, Vol. 164, No. 1, pp. 54-64, 1996.
9. T. Pedersen. A threshold cryptosystem without a trusted party. In Advances in Cryptology-EUROCRYPT'91, LNCS 547, Springer-Verlag, pp. 522-536, 1991.
10. T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Advances in Cryptology-CRYPT'01, LNCS 576, Springer-Verlag, pp. 129-140, 1991.
11. A. Shamir. How to Share a Secret. Communications of the ACM, 22:612-613, 1979.
12. V. Shoup and R. Gennaro, Securing Threshold Cryptosystems against Chosen Ciphertext Attack, Journal of Cryptology, Vol. 15, Springer-Verlag 2002, pages 75-96.
13. H. Wang, Y. Zhang, and D. Feng. Short Threshold Signature Schemes Without Random Oracles. INDOCRYPT 2005, LNCS 3797, Springer-Verlag, pp. 297-310, 2005.
14. B.Waters, Efficient Identity based Encryption without random oracles. In Advances in Cryptology-EuroCrypt'05, LNCS 3494, Springer-Verlag, pp. 114-127, 2005.