

A Forward Secure Threshold Signature Scheme Based on the Structure of Binary Tree

Jia Yu¹

¹College of Information Engineering, Qingdao University, Qingdao, P. R. China
Email: qduyujia@gmail.com

Fanyu Kong², Xiangguo Cheng¹, Rong Hao¹

²Institute of Network Security, Shandong University, Jinan, P. R. China
Email: {sdukongfanyu, xiangguocheng, hr}@gmail.com

Abstract—Forward secure threshold signature plays an important role in distributed signature. Based on binary tree structure, a new forward secure threshold signature from bilinear pairings is proposed in this paper. In this scheme, each cost of key generation algorithm, key update algorithm, signing algorithm and verifying algorithm is independent of the total number of time periods. At the same time, the scheme needs very few interactions. Because the bilinear pairing used in this scheme is operating over a certain elliptic curve, the scheme inherits the property of short signature, that is, it has short secret key, public key and signature. We formalize the definition of the security model of forward secure threshold signature and prove the proposed scheme is forward secure under the computation Diffie-Hellman assumption in the random oracle model.

Index Terms—forward security, bilinear pairings, threshold cryptography

I. INTRODUCTION

Forward secure threshold signature is one kind of important distributed signatures. In an ordinary threshold signature, the signing secret key is divided into several pieces called shares that are held by multiple players, respectively. And only no fewer than a quorum number of players can cooperate to produce the signature. Therefore, threshold signature can make the secret key exposure more difficultly. The forward secure threshold signature cannot only make the key exposure difficultly but also reduce the damage of secret key exposure. In this paradigm, the whole lifetime of signature is divided into multiple time periods. In each period, all players update their secret shares to make the corresponding secret key evolve, however, the public key is fixed during the whole lifetime. Forward secure threshold signature satisfies: If an adversary corrupts fewer than a quorum number of players, she cannot forge signatures of any time periods; even if an adversary can corrupt a quorum number of players to get threshold shares in a certain period, she cannot forge any signature of any previous period.

Threshold signature scheme was firstly presented in Ref. [1]. Afterwards, a lot of related works were done such as [2, 3]. Anderson [4] firstly proposed forward

security for digital signature. Refs. [5~13] presented various forward secure signatures with different properties. Abdalla *et al.* [14] proposed the first forward secure threshold signature based on scheme [5]. Unfortunately, the public key size and the secret key size are very large in this scheme. What's more, this scheme needed many interactions due to using distributed multiplication of many values protocol. And then, another forward secure threshold signature with proactive property [15] was presented, which was based on scheme [6]. This scheme used a shorter secret key, but had lower efficiency. Wang *et al.* [16] pointed out the distributed multiplication protocol in scheme [15] was insecure. Chu *et al.* [17] proposed a forward secure threshold signature scheme that couldn't tolerate malicious adversary and had not any security proofs as an extension of his main work. Recently, Ref. [18] proposed a forward secure threshold signature from bilinear pairings based on scheme [10].

In this paper, we propose a new forward secure threshold signature scheme from bilinear pairings based on [11]. Because the proposed scheme adopts binary tree structure to perform key storage and key update, it makes the sub-algorithms very efficient. Different from previous schemes, the costs of key generation, key update, signing and verifying algorithms are all independent of the total time period T . It means our scheme will still very efficient when T is a large number, which is impossible for all previous schemes. Because the scheme is constructed over a certain elliptic curve, it has shorter public key, secret key and signature. Furthermore, we give the formal security definition of forward secure threshold signature. Finally, we prove that the scheme is forward secure under the computation Diffie-Hellman assumption in the random oracle model.

II. PRELIMINARIES

A. CDH assumption and Bilinear Pairing

Let G_1 be an additive group of prime order q and G_2 be a multiplicative group of the same prime order q . And $P \in G_1$ is a generator of group G_1 .

--Computation Diffie-Hellman problem (CDHP):
Given (P, aP, bP) where $a, b \in_R Z_q$, compute abP .

Definition 1 (CDH assumption). A probabilistic algorithm A is said (t, ϵ) -break CDHP in G_1 if A runs at most time t , computes CDHP with an advantage of at least ϵ . We say that G_1 is a (t, ϵ) -break CDH group if no probabilistic algorithm A (t, ϵ) -break CDHP in G_1 .

A map $\hat{e}: G_1 \times G_1 \rightarrow G_2$ is called a bilinear pairing if following properties are satisfied:

1. Bilinear: For all $P, Q \in G_1$ and $a, b \in Z$, there is $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$.
2. Non-degenerate: The map does not send all pairs in $G_1 \times G_1$ to the identity in G_2 .
3. Computable: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in G_1$.

A randomized algorithm IG that takes as input a security parameter $k \in Z$ is a CDH parameter generator if it runs in time polynomial in k and outputs the description of two groups G_1, G_2 and a bilinear map $\hat{e}: G_1 \times G_1 \rightarrow G_2$. We denote the output of this algorithm as $(G_1, G_2, \hat{e}) = IG(1^k)$.

B. Forward Secure Threshold Signature and Its Security

Definition 2 (Key-evolving threshold signature). A key-evolving threshold signature scheme is a quadruple of algorithms, $FTS(t, s, n) = (FTS.key, FTS.update, FTS.sign, FTS.verify)$, where t is the maximum number of players corrupted by the adversary; s is the minimum number of honest players so that signature computation is feasible; n is the total number of players.

FTS.key: the key generation algorithm, inputs a security parameter $k \in N$ and the total number of time periods T , and generates a public key PK and the initial shares $SK_0^{(i)}$ of the secret key for player $i (i=1, 2, \dots, n)$.

FTS.update: the secret key update algorithm, inputs the current time period j , and generates $SK_{j+1}^{(i)}$ for each player i for the next time period by a distributed protocol.

FTS.sign: the signing algorithm, inputs the current time period j and a message M , and the participant players jointly generate a signature $\langle j, \text{tag} \rangle$ of message M for period j using their shares.

FTS.verify: the verification algorithm, inputs the public key PK , a message M and a signature $\langle j, \text{tag} \rangle$, and returns 1 if $\langle j, \text{tag} \rangle$ is a valid signature of M or 0, otherwise.

We say that $\langle j, \text{tag} \rangle$ is a valid signature of message M if $FTS.verify(M, \langle j, \text{tag} \rangle) = 1$.

If a key-evolving threshold signature scheme is a forward secure threshold signature scheme, it needs to satisfy: even if the adversary corrupts up to t players, it is computationally infeasible for her to forge any signature of previous time period. We give the experiment to evaluate the security in random oracle (RO) model [19]:

Experiment F-Forge-RO(FTS, F)

Randomly select $H: \{0, 1\}^* \rightarrow \{0, 1\}^l$

$(PK, SK_0^{(1)}, \dots, SK_0^{(n)}) \leftarrow^{R} FTS.key^H(k, \dots, T)$

Repeat

$d \leftarrow F^{H, FTS.sign_{SK_j^{(i)}}}(\text{cma}, PK)$;

$SK_{j+1}^{(i,j,1)} \leftarrow FTS.update^H(SK_j^{(i,1)}, SK_j^{(i,2)}, \dots, SK_j^{(i,k)})$,

$k \geq t+1$;

$j \leftarrow j+1$;

Until $(d = \text{overthreshold}) \text{ or } (j = T)$

$b \leftarrow j$

$(M, \langle i, \text{sign} \rangle) \leftarrow F^H(\text{forge}, SK_j^{(i,1)}, SK_j^{(i,2)}, \dots, SK_j^{(i,k)})$,

$k \geq t+1$;

if $FTS.verify_{PK}^H(M, \langle i, \text{sign} \rangle) = 1$ and $1 \leq i < b$
and M was not queried of $FTS.sign_{SK_i}^H(\cdot)$

in period i

then return 1

else return 0

From above experiment, we can obtain the adversary model in RO model: the adversary knows the public key, the total number of time periods T and the current time period. A hash function H is viewed as a random oracle. Adversary F runs in three phases: in the first phase, chosen message attack phase (cma), F can query the signature of any message she selects with respect to the current secret key by accessing to a signature oracle. At the end of each time period, she can decide whether to stay in this phase or switch to over-threshold phase. In the second phase, over-threshold phase, for a particular time period b , the adversary may corrupt up to a threshold number of players. It means F can learn the secret key SK_b . In the last phase, the forgery phase, the adversary outputs a signature message pair, that is, a forgery. The adversary is considered to be successful if she forges a signature of some new message (that is, not queried previously) for some time period prior to b . During the whole procedure, F can query the random oracle H corresponding to a collision-resistant hash function. Depending on the verification results, the experiment will return 1 or 0 to indicate the success or failure of the adversary F .

Definition 2 (Forward-security in the Random Oracle Model). Let $FTS(t, s, n) = (FTS.key, FTS.update, FTS.sign, FTS.verify)$ be a key-evolving threshold signature scheme, H be a random oracle and the algorithm F be an adversary as described above. We say that an algorithm $F(t, q_s, q_H, \epsilon)$ -attack FTS if F runs in time at most t and makes at most q_s signing queries to the signature oracle and q_H hash queries to the H oracle, and then $Adv_{FTS, A} \geq \epsilon$. We say that FTS is (t, q_s, q_H, ϵ) -forward secure against chosen message attacks in the random oracle if there is no adversary $F(t, q_s, q_H, \epsilon)$ -attack FTS , and denote it by $FTS(t, q_s, q_H, \epsilon)$.

C. Building Blocks

- (1) Verifiably distributed secret generation protocol (VDSG)

All players jointly and verifiably generate a random secret ρ . Finally, each player i holds a secret share ρ_i and is able to verify whether her share ρ_i is valid or not. The public commits include ρP_0 and $\rho_i P_0 (i=1..n)$.

We use the Joint-Exp-RSS protocol [20] as the VDSG protocol in our scheme. We will use the security results of it to prove the security of our threshold signature scheme.

- (2) Zero-knowledge proof protocol [18]

Let G be a cyclic group of some prime order q , where G is represented additively. Let $P_i (i=0..n)$ be the generators of G .

Prover P wants to convince verifier V that she knows these values $b_i (i=1..n)$ that satisfy

$$G_i = b_i P_0 (i=1..n), H' = \sum_{i=1}^n b_i P_i.$$

We give a non-interactive version by a collision-resistant hash function: $H: G \rightarrow Z_q$ in [18]. The protocol NI Proof-VS($P_0; P_1, \dots, P_n; G_1, \dots, G_n; H'$) is described as follows:

- ① P selects $w_i \in_R Z_q (i=1..n)$ at random, and computes $E_i = w_i P_0 (i=1..n)$, $F = \sum_{i=1}^n w_i P_i$, $c = H(P_0 \parallel P_1 \parallel \dots \parallel P_n \parallel G_1 \parallel \dots \parallel G_n \parallel H' \parallel E_1 \parallel \dots \parallel E_n \parallel F)$, and $r_i = w_i - b_i c, (i=1..n)$. Then sends $c, r_i (i=1..n)$ to V .

- ② V verifies:

$$c \stackrel{?}{=} H(P_0 \parallel P_1 \parallel \dots \parallel P_n \parallel G_1 \parallel \dots \parallel G_n \parallel H' \parallel r_1 P_0 + c G_1 \parallel \dots \parallel r_n P_0 + c G_n \parallel c H' + \sum_{i=1}^n r_i P_i)$$

If the equation is right, V believes P ; otherwise, doesn't.

III. THE PROPOSED FORWARD SECURE THRESHOLD SIGNATURE

A. The Binary Tree Structure and Notations

The scheme adopts binary tree structure that has been used in many cryptographic designs such as hierarchical ID-based cryptography [21]. A full binary tree with depth l can represent $T = 2^{l+1} - 1$ time periods. Each node of the tree is associated with one time period. Let $w^0 = \varepsilon$, where ε denotes an empty string. Let w^j denote the node associated with period j . Let $w^j 0 (w^j 1)$ be the left (right) child node of w^j , $w^j|_k$ be a k -prefix of w^j . Associate all nodes of the tree with the time periods according to the pre-order traversal: Begin with root node w^0 . If w^j is an internal node, then $w^{j+1} = w^j 0$, if w^j is a leaf node and $j < T - 1$, then $w^{j+1} = w^j 1$, where w^j is the longest string such that $w^j 0$ is a prefix of w^j .

The secret share $SK_j^{(i)}$ player i holds in period j is a set which is composed of the node secret share $S_{w^j}^{(i)}$ and the secret shares of the right siblings of the nodes on the path from the root to w^j . That is, whenever $w^j 0$ is a prefix of w^j , $SK_j^{(i)}$ contains the share $S_{w^j 1}^{(i)}$ of secret key of node $w^j 1$. The secret share $SK_j^{(i)}$ is organized as a stack $ST^{(i)}$ of the shares of node secrets when player i runs the key update algorithm at the end of period j . At that time $S_{w^j}^{(i)}$ lies in the top of $ST^{(i)}$. Firstly pop the current node secret share $S_{w^j}^{(i)}$ off the stack, then update as follows:

- 1. If w^j is an internal node, generate the secret shares $S_{w^j 0}^{(i)}$ and $S_{w^j 1}^{(i)}$ of $w^j 0$ and $w^j 1$, respectively. And then push $S_{w^j 1}^{(i)}$ and $S_{w^j 0}^{(i)}$ onto the stack orderly. The new top is $S_{w^j 0}^{(i)}$ and indeed $w^{j+1} = w^j 0$. Erase $S_{w^j}^{(i)}$ at last.
- 2. If w^j is a leaf, erase $S_{w^j}^{(i)}$. The next share on top of the stack is $S_{w^{j+1}}^{(i)}$.

B. Description of The Scheme

- (1) FTS.key: Input a security parameter k and the depth l of a binary tree. Do as follows:

- ① Run $IG(1^k)$ to generate groups G_1 and G_2 of some prime order q and an admissible pairing $\hat{e}: G_1 \times G_1 \rightarrow G_2$.
- ② Select a generator $P \in_R G_1$, random value $\rho_\varepsilon \in_R Z_q$ and set $R = \rho_\varepsilon P$. Select $a_i \in_R Z_q (i=1..t)$ and set $f(x) = \rho_\varepsilon + \sum_{i=1}^t a_i x^i \pmod{q}$. Compute $\rho_\varepsilon^{(i)} = f(i), (i=1..n)$.
- ③ Select cryptographic hash functions $H_1: \{0,1\}^* \times G_1 \rightarrow G_1$, $H_2: \{0,1\}^* \times G_1 \times G_1 \rightarrow G_1$, $H_3: \{0,1\}^* \times G_1 \rightarrow Z_q^*$, $H_4: G_1 \times \{0,1\}^* \times G_1 \rightarrow Z_q^*$.

- ④ Let the public key $PK = (G_1, G_2, e, P, R, l, H_1, H_2, H_3, H_4)$. Compute and broadcast $R_\varepsilon^{(i)} = \rho_\varepsilon^{(i)} P (i=1..n)$. Send $\rho_\varepsilon^{(i)}$ to player $i (i=1..n)$ secretly. Players $i (i=1..n)$ compute $SN_\varepsilon^{(i)} = \rho_\varepsilon^{(i)} H_1(\varepsilon, R)$. Set the root node secret share $S_\varepsilon^{(i)} = (Z_\varepsilon = 0 \in G_1, SN_\varepsilon^{(i)})$ and set initial secret shares $SK_0^{(i)} = (S_\varepsilon^{(i)})$ and push it onto the stack. $ST^{(i)}$.

- (2) FTS.update: Input the public key PK , time period j and the secret shares $SK_{w^j}^{(i)}$. Firstly, each player $i (i=1..n)$ pops the node secret share $S_{w^j}^{(i)} = (Z_{w^j}, SN_{w^j}^{(i)})$ off the stack $ST^{(i)} = SK_k^{(i)}$, and then does as follows:

- ① If w^j is an internal node, all players jointly generate two random values $\rho_{w^j 0}, \rho_{w^j 1} \in Z_q$ by executing twice *VDSG* protocol simultaneously. Player i gets shares $\rho_{w^j 0}^{(i)}, \rho_{w^j 1}^{(i)} \in_R Z_q$ and public commits
- $$R_{w^j 0}^{(i)} = \rho_{w^j 0}^{(i)} P, R_{w^j 1}^{(i)} = \rho_{w^j 1}^{(i)} P \text{ and } R_{w^j 0} = \rho_{w^j 0} P, R_{w^j 1} = \rho_{w^j 1} P.$$
- Player i firstly computes $h_{w^j 0} = H_3(w^j 0, R_{w^j 0}), h_{w^j 1} = H_3(w^j 1, R_{w^j 1})$; then computes $Z_{w^j 0} = Z_{w^j} + h_{w^j 0} R_{w^j 0}, Z_{w^j 1} = Z_{w^j} + h_{w^j 1} R_{w^j 1}$; at last computes $SN_{w^j 0}^{(i)} = SN_{w^j}^{(i)} + \rho_{w^j 0}^{(i)} h_{w^j 0} H_1(\epsilon, R), SN_{w^j 1}^{(i)} = SN_{w^j}^{(i)} + \rho_{w^j 1}^{(i)} h_{w^j 1} H_1(\epsilon, R)$. Player i erases $S_{w^j}^{(i)}$ and pushes $S_{w^j 1}^{(i)} = (Z_{w^j 1}, SN_{w^j 1}^{(i)}), S_{w^j 0}^{(i)} = (Z_{w^j 0}, SN_{w^j 0}^{(i)})$ onto the stack $ST^{(i)}$ orderly. At that time, the top element in the stack is $S_{w^j 0}^{(i)}$.
- ② If w^j is a leaf, then directly erases $S_w^{(i)}$. At that time, the top element in the stack is $S_{w^{j+1}}^{(i)}$.

(3) *FTS.sign*: Input a message M and the current time period j . Let $w^j = w_1 \dots w_t$ denote the node corresponding to period j .

- ① Each player i reads the node secret share $S_{w^j}^{(i)} = (Z_{w^j}, SN_{w^j}^{(i)})$ from the top of the stack $ST^{(i)}$.
- ② All players jointly generate a random secret $r \in Z_q$ by executing *VDSG* protocol. Player i gets the share $r^{(i)} \in Z_q$ and the public commits $U = rP, U^{(j)} = r^{(j)}P$, where $j = 1, \dots, n$.
- ③ Player i computes partial signature: $FS^{(i)} = SN_{w^j}^{(i)} + r^{(i)} H_2(w^j || M, U, Z_{w^j})$ and executes *NI Proof-VS* ($P; H_1(\epsilon, R), h_{w^j l} H_1(\epsilon, R), \dots, h_{w^j t} H_1(\epsilon, R), H_2(w^j || M, U, Z_{w^j}); R_\epsilon^{(i)}, R_{w^j 1}^{(i)}, \dots, R_{w^j t}^{(i)}, U^{(i)}$); $FS^{(i)}$ to prove the part signature $FS^{(i)}$ which she provides satisfies $FS^{(i)} = \rho_\epsilon^{(i)} H_1(\epsilon, R) + \sum_{m=1}^t h_{w^j m} \rho_{w^j m}^{(i)} H_1(\epsilon, R) + r^{(i)} H_2(w^j || M, U, Z_{w^j})$, and these $\rho_\epsilon^{(i)}, \rho_{w^j m}^{(i)}, (m=1..t)$ and $r^{(i)}$ satisfies: $R_\epsilon^{(i)} = \rho_\epsilon^{(i)} P, R_{w^j m}^{(i)} = \rho_{w^j m}^{(i)} P (m=1, \dots, t), U^{(i)} = r^{(i)} P$. If these verifications pass, it means the player i provides a valid partial signature.
- ④ Any set B of $t+1$ players who pass the verification compute $FS = \sum_{i \in B} C_{B_i} FS^{(i)}$.
- ⑤ Output the signature $\langle j, \sigma = (U, Z_{w^j}, FS) \rangle$.

(4) *FTS.verify*: Input a signature $\langle j, \sigma = (U, Z_{w^j}, FS) \rangle$ in period j for a message M . Verify the following equation holds or not:

$$\hat{e}(P, FS + H_4(U, w^j, Z_{w^j}) \cdot H_1(\epsilon, R)) \stackrel{?}{=} \hat{e}(R + Z_{w^j} + H_4(U, w^j, Z_{w^j})P, H_1(\epsilon, R)) \cdot \hat{e}(U, H_2(w^j || M, U, Z_{w^j}))$$

If it holds return 1; else return 0.

IV. PERFORMANCE COMPARISONS

The complexity analysis is considered in terms of T like [9,10]. The table 1 gives the comparisons among our scheme, schemes in [14,15,18], where l' is a security parameter in scheme [14,15]. The complexities of key generation, key update, signing and verifying algorithms in scheme [14] and scheme [15] are $O(1)T$ and $O(1)l'T$, respectively. The complexities of key generation, key update algorithms are $O(1)$, and the complexities of signing and verifying algorithms are $O(1)\log T$ in scheme [18]. Thanks to the pre-order traversal technique of binary trees, the operations of key generation and key update algorithms are independent of the total number of time periods T in our proposed *FTS* scheme. The complexities of signing and verifying algorithms are both $O(1)$ due to the adopted new strategy in key update.

The total interactions in our scheme are very few. There is no interaction in our key generation algorithm. Key update algorithm will execute twice *VDSG* protocol simultaneously, but only needs once interaction. In signing algorithm twice interactions are needed in total, one happens in *VDSG* protocol and the other happens in *NI Proof-VS* protocol.

TABLE I. PERFORMANCE COMPARISONS

	Scheme in [14]	Scheme in [15]	Scheme in [18]	Our scheme
<i>FTS.key</i> time and interactions	$O(1)T$	$O(1)l'T$	$O(1)$	$O(1)$
	0	1	0	0
<i>FTS.update</i> time and interactions	$O(1)T$	$O(1)l'T$	$O(1)$	$O(1)$
	1	2	1	1
<i>FTS.sign</i> time and interactions	$O(1)T$	$O(1)l'T$	$O(1)\log T$	$O(1)$
	$2l'$	2	2	2
<i>FTS.verify</i> time and interactions	$O(1)T$	$O(1)l'T$	$O(1)\log T$	$O(1)$
	0	0	0	0

V. SECURITY ANALYSIS

Theorem 1. Let $PK = (G_1, G_2, e, P, R, l, H_1, H_2, H_3, H_4)$ and $SK_0^{(i)} = S_\epsilon^{(i)} = (Z_\epsilon, SN_\epsilon^{(i)})$ be the public key and the secret shares of player $i (i=1,2,\dots,n)$ generated by algorithm *FTS.key*, respectively; Let the shares of secret key be updated by algorithm *FTS.update*; Let $\langle j, \sigma = (U, Z_{w^j}, FS) \rangle$ be the signature in period j for

message M generated by algorithm $FTS.sign$. Then $FTS.verify(M, < j, \sigma = (U, Z_{w^j}, FS) >) = 1$.

Proof.

$$\begin{aligned}
 & \hat{e}(R + Z_{w^j} + H_4(U, w^j, Z_{w^j})P, H_1(\varepsilon, R)) \\
 & \cdot \hat{e}(U, H_2(w^j \| M, U, Z_{w^j})) \\
 = & \hat{e}(\rho_\varepsilon P + \sum_{m=1}^t h_{w^j|_m} \rho_{w^j|_m} P + H_4(U, w^j, Z_{w^j})P, H_1(\varepsilon, R)) \\
 & \cdot \hat{e}(rP, H_2(w^j \| M, U, Z_{w^j})) \\
 = & \hat{e}(P, (\rho_\varepsilon + \sum_{m=1}^t h_{w^j|_m} \rho_{w^j|_m} + H_4(U, w^j, Z_{w^j}))H_1(\varepsilon, R)) \\
 & \cdot \hat{e}(P, rH_2(w^j \| M, U, Z_{w^j})) \\
 = & \hat{e}(P, (\rho_\varepsilon + \sum_{m=1}^t h_{w^j|_m} \rho_{w^j|_m}) \cdot H_1(\varepsilon, R) \\
 & + H_4(U, w^j, Z_{w^j}) \cdot H_1(\varepsilon, R) + rH_2(w^j \| M, U, Z_{w^j})) \\
 = & \hat{e}(P, (H_4(U, w^j, Z_{w^j}) + \sum_{i \in B} C_{Bi}(\rho_\varepsilon^{(i)} + \sum_{m=1}^t h_{w^j|_m} \rho_{w^j|_m}^{(i)})) \\
 & \cdot H_1(\varepsilon, R) + \sum_{i \in B} C_{Bi} r^{(i)} H_2(w^j \| M, U, Z_{w^j})) \\
 = & \hat{e}(P, \sum_{i \in B} C_{Bi} (SN_{w^j}^{(i)} + r^{(i)} H_2(w^j \| M, U, Z_{w^j})) \\
 & + H_4(U, w^j, Z_{w^j}) \cdot H_1(\varepsilon, R)) \\
 = & \hat{e}(P, \sum_{i \in B} C_{Bi} FS^{(i)} + H_4(U, w^j, Z_{w^j}) \cdot H_1(\varepsilon, R)) \\
 = & \hat{e}(P, FS + H_4(U, w^j, Z_{w^j}) \cdot H_1(\varepsilon, R))
 \end{aligned}$$

Theorem 2. When $s \geq t+1$ and $n \geq 2t+1$, our $FTS(t, s, n)$ scheme can tolerate an adversary able to corrupt t players.

Proof.

When $s \geq t+1$ and $n \geq 2t+1$, even if the adversary is able to corrupt t players, there are still $s \geq t+1$ honest players. These honest players can make $FTS.update$ and $FTS.sign$ algorithms be executed properly. According to theorem 2, the scheme can tolerate a malicious adversary corrupting t players.

Theorem 3. If the group G_1 generated by $IG(1^k)$ in $FTS.key(k, l)$ is a (t', ε') -break CDHP group, then the $FTS(t, s, n)$ scheme we propose is a $(t, q_S, q_{H_2}, \varepsilon)$ -forward secure against chosen message attacks in the random oracle.

$$\text{where } t = t' - O(T \cdot k^n), \quad \varepsilon = T\varepsilon' + \frac{(q_{H_2} - 1) \cdot q_S}{q - 1}.$$

Proof. Similarly to the method in Ref. [10], we can replace the hash functions H_1 and H_3 with 1-wise and $(l+1)$ -wise independent hash functions in function families. We view H_2 as a random oracle and H_4 as an ordinary hash function in the following proof. Assuming F being an adversary $(t, q_S, q_{H_2}, \varepsilon)$ -attack $FTS(t, s, n)$, we construct a PPT adversary I (t', ε') -break CDHP in group G_1 .

Firstly, the algorithm I is given parameters (G_1, G_2, \hat{e}) generated by $IG(1^k)$ and a challenge $(P, R = \alpha P, \beta P)$, and the goal of I is to compute $\alpha\beta P$, where $\alpha = \rho_\varepsilon$ and $\beta \in_R Z_q^*$ are unknown to I . I runs F as a subroutine. I selects a total time periods T and guesses the time period

b randomly at which F asks the over-threshold queries, where $0 < b \leq T$. Let $w^b = w_1^* \dots w_s^*$ denote the node corresponding to period b . I chooses $r_{w^b}, h_{w^b} \in_R Z_q^*$, and chooses $r_{w^b|_i}, h_{w^b|_i} \in_R Z_q^*$ for all $1 \leq i \leq s$ and $w_i^* = 0$. I randomly selects hash function H_1 and H_3 from 1-wise and $(l+1)$ -wise independent hash families, respectively, with the following constraints:

$$\begin{aligned}
 H_1(\varepsilon, R) &= \beta P = I_\varepsilon, \\
 R_{w^b} &= 1/h_{w^b}(r_{w^b}P - R), \\
 H_3(w^b, R_{w^b}) &= h_{w^b}, \\
 \text{For all } 1 \leq i \leq s \text{ and } w_i^* &= 0: \\
 R_{w^b|_i} &= 1/h_{w^b|_i}(r_{w^b|_i}P - R), \quad H_3(w^b|_i, R_{w^b|_i}) = h_{w^b|_i}.
 \end{aligned}$$

I provides $PK = (G_1, G_2, e, P, R, l, H_1, H_2, H_3, H_4)$ and T to F . I maintains two tables: H_2 oracle table and signature query table to answer the queries from F .

I simulates the $FTS.update$ procedure at first in order to provide necessary parameters for replying to F 's signature queries and over-threshold query. Let $w^j = w_1 \dots w_t$ denote the node corresponding to period j . For all $j = 0, \dots, b-1$, I simulates $FTS.update$ procedure orderly as follows:

1. If w^j is a leaf, then I does nothing.
2. If $w^j 0 = w^b$, then according to $r_{w^j 0}, r_{w^j 1}, h_{w^j 0}, h_{w^j 1} \in Z_q^*$ which have been defined during selecting H_3 , set $R_{w^j 0} = 1/h_{w^j 0}(r_{w^j 0}P - R)$, $H_3(w^j 0, R_{w^j 0}) = h_{w^j 0}$, $R_{w^j 1} = 1/h_{w^j 1}(r_{w^j 1}P - R)$, $H_3(w^j 1, R_{w^j 1}) = h_{w^j 1}$, and computes $Z_{w^j 0} = Z_{w^j 0} + h_{w^j 0}R_{w^j 0}$, $Z_{w^j 1} = Z_{w^j 1} + h_{w^j 1}R_{w^j 1}$.
3. If $w^j 0 \neq w^b$ is a prefix of w^b , then selects $\rho_{w^j 0}, h_{w^j 0} \in_R Z_q^*$, and sets $R_{w^j 0} = \rho_{w^j 0}P$, $H_3(w^j 0, R_{w^j 0}) = h_{w^j 0}$. According to $r_{w^j 1}, h_{w^j 1} \in Z_q^*$ which have been defined during selecting H_3 , sets $R_{w^j 1} = 1/h_{w^j 1}(r_{w^j 1}P - R)$, $H_3(w^j 1, R_{w^j 1}) = h_{w^j 1}$, and computes $Z_{w^j 0} = Z_{w^j 0} + h_{w^j 0}R_{w^j 0}$, $Z_{w^j 1} = Z_{w^j 1} + h_{w^j 1}R_{w^j 1}$.
4. Otherwise, selects $\rho_{w^j 0}, h_{w^j 0}, \rho_{w^j 1}, h_{w^j 1} \in_R Z_q^*$, and sets $R_{w^j 0} = \rho_{w^j 0}P$, $H_3(w^j 0, R_{w^j 0}) = h_{w^j 0}$, $R_{w^j 1} = \rho_{w^j 1}P$, $H_3(w^j 1, R_{w^j 1}) = h_{w^j 1}$. I computes $Z_{w^j 0} = Z_{w^j 0} + h_{w^j 0}R_{w^j 0}$, $Z_{w^j 1} = Z_{w^j 1} + h_{w^j 1}R_{w^j 1}$.

At that time, F runs in cma phase. F may query H_2 oracle and signature oracle, so I needs to simulate these oracles to answer the queries. In doing so, we have to simulate F 's view $VIEW_F$ of the protocol. W.l.o.g. assume that the adversary F corrupts players $1, 2, \dots, t$.

The simulation of H_2 queries: When F queries the oracle H_2 at a point $\langle P_M, U, Z_{w^j} \rangle$ where $P_M = w^j \| M$, I does as follows:

1. If $\langle P_M, U, Z_{w^j} \rangle$ has already appeared on a tuple $\langle P_M, U, Z_{w^j}, h, \gamma, \phi \rangle$ in H_2 table, then I responds $H_2(P_M, U, Z_{w^j}) = h \in G_1$ to F .
2. Else selects $x_{w^j} \in_R Z_q$ and adds $\langle P_M, U, Z_{w^j}, h = \gamma P, \gamma, * \rangle$ to H_2 table. I responds $H_2(M, j, U) = h \in G_1$ to F .

The simulation of signature oracle queries: When F queries the signature at a point $\langle M, j \rangle$, I does as follows:

1. Selects $\gamma, \phi \in_R Z_q^*$, and sets $h = \gamma P - I_\epsilon / \phi$, $U = \phi R$ ($r = \phi \alpha$). If $\langle w^j \parallel M, U, Z_{w^j} \rangle$ has appeared in H_2 table, then I aborts.
2. Adds $\langle w^j \parallel M, U, Z_{w^j}, h, \gamma, \phi \rangle$ to H_2 table.
3. I uses $\rho_{w^j|_m}, h_{w^j|_m}$ ($1 \leq m \leq t$) generated during simulating $F.TS.update$ procedure to compute

$$FS = \sum_{m=1}^t \rho_{w^j|_m} h_{w^j|_m} I_\epsilon + \phi \gamma R + H_4(U, w^j, Z_{w^j}) \cdot I_\epsilon$$

Since

$$\begin{aligned} FS &= \alpha I_\epsilon + \sum_{m=1}^t \rho_{w^j|_m} h_{w^j|_m} I_\epsilon \\ &\quad + r \cdot H_2(w^j \parallel M, U, Z_{w^j}) + H_4(U, w^j, Z_{w^j}) \cdot I_\epsilon \\ &= \alpha I_\epsilon + \sum_{m=1}^t \rho_{w^j|_m} h_{w^j|_m} I_\epsilon + \phi \alpha \cdot (\gamma P - I_\epsilon / \phi) \\ &\quad + H_4(U, w^j, Z_{w^j}) \cdot I_\epsilon \\ &= \sum_{m=1}^t \rho_{w^j|_m} h_{w^j|_m} I_\epsilon + \phi \gamma R + H_4(U, w^j, Z_{w^j}) \cdot I_\epsilon \end{aligned}$$

4. According to Z_{w^j} generated during simulating $F.TS.update$ procedure, I responds $\langle j, \sigma = (U, Z_{w^j}, FS) \rangle$ to F .

Obviously, I can provide the signature to F though she can't compute $\alpha I_\epsilon = \alpha \beta P$.

The simulation of $VIEW_F$ in $F.TS.key$: Because $f(x)$ is a random polynomial in Z_q , and α_i is a random value, $SN_\epsilon^{(i)}$ is distributed uniformly in G_1 . We can pick values for α_i ($i=1..t$) at random from Z_q . And then compute $SN_\epsilon^{(i)}, R_\epsilon^{(i)}$ ($i=1..t$) and Z_ϵ . For each $R_\epsilon^{(j)}$ ($j=t+1..n$), compute

$$\begin{aligned} R_\epsilon^{(j)} &= \alpha_j P = (\lambda_{j,0} \cdot \alpha + \sum_{i=1}^t \lambda_{j,i} \cdot \alpha_i) P \\ &= \lambda_{j,0} Q + \sum_{i=1}^t \lambda_{j,i} \cdot R_\epsilon^{(i)}, \end{aligned}$$

where $\lambda_{j,i}$ are computable Lagrange interpolation coefficients. The simulation of $VIEW_F$ in $F.TS.key$ has been completed.

The simulation of $VIEW_F$ in $F.TS.update$: Because the shares of secrets ρ_{w_0}, ρ_{w_1} are distributed uniformly in Z_q , we can pick random values $\rho_{w_0}^{(i)}, \rho_{w_1}^{(i)}$ ($i=1..t$) in Z_q for F . It is easy to compute $SN_{w_1}^{(i)}, SN_{w_0}^{(i)}$ and provide them to F . According to the security proof of the $VDSG$ protocol, taking as input R_{w_0}, R_{w_1} , we can simulate the $VDSG$

protocol to get $VIEW_F$ including $R_{w_0}^{(i)}, R_{w_1}^{(i)}$ ($i=1..n$) in this protocol.

The simulation of $VIEW_F$ in $F.TS.sign$: For a message $\langle M, k \rangle$, we take U got from query signature oracle as input to simulate the $VDSG$ protocol, therefore, we can get $VIEW_F$ in the protocol including $U^{(i)} = r^{(i)} P$ ($i=1..n$). The value of $H_2(M, k, U)$ can be obtained by H_2 hash oracle query and FS_i ($i=1..t$) can be computed according to r_i ($i=1..t$) and $SN_w^{(i)}$ ($i=1..t$) from simulation of $F.TS.update$. With FS obtained from the query of the signature oracle, we can compute FS_i ($i=t+1..n$) which F views by the means of Lagrange interpolation in simulation of $F.TS.sign$. Simulate the $VDSG$ protocol at last.

From above description, we can know that the $VIEW_F$ F gets from the protocol can be simulated successfully.

When F finishes the cma phase and comes to the over-threshold phase in period b , I does as follows in order to provide SK_b to F :

According to the parameters generated during simulating $F.TS.update$ procedure, I computes

$$S_{w^b} = r_{w^b} I_\epsilon + \sum_{m=1}^{s-1} \rho_{w^b|_m} h_{w^b|_m} I_\epsilon$$

Since

$$\begin{aligned} S_{w^b} &= \alpha I_\epsilon + \rho_{w^b} h_{w^b} I_\epsilon + \sum_{m=1}^{s-1} \rho_{w^b|_m} h_{w^b|_m} I_\epsilon \\ &= \alpha I_\epsilon + 1/h_{w^b} (r_{w^b} - \alpha) h_{w^b} I_\epsilon + \sum_{m=1}^{s-1} \rho_{w^b|_m} h_{w^b|_m} I_\epsilon \\ &= r_{w^b} I_\epsilon + \sum_{m=1}^{s-1} \rho_{w^b|_m} h_{w^b|_m} I_\epsilon \end{aligned}$$

For all the nodes $w^b|_i$ ($1 \leq i \leq s$) satisfying $w_i^* = 0$ on the path from the root to w^b , I computes the node secret keys $S_{w^b|_i} = r_{w^b|_i} I_\epsilon + \sum_{m=1}^{i-1} \rho_{w^b|_m} h_{w^b|_m} I_\epsilon$ for their right siblings $w^b|_{\bar{i}}$. Since

$$\begin{aligned} S_{w^b|_{\bar{i}}} &= \alpha I_\epsilon + \rho_{w^b|_{\bar{i}}} h_{w^b|_{\bar{i}}} I_\epsilon + \sum_{m=1}^{i-1} \rho_{w^b|_m} h_{w^b|_m} I_\epsilon \\ &= \alpha I_\epsilon + 1/h_{w^b|_{\bar{i}}} (r_{w^b|_{\bar{i}}} - \alpha) h_{w^b|_{\bar{i}}} I_\epsilon + \sum_{m=1}^{i-1} \rho_{w^b|_m} h_{w^b|_m} I_\epsilon \\ &= r_{w^b|_{\bar{i}}} I_\epsilon + \sum_{m=1}^{i-1} \rho_{w^b|_m} h_{w^b|_m} I_\epsilon \end{aligned}$$

The values $Z_{w^b|_i}$ ($1 \leq i \leq s$) and Z_{w^b} have been computed when I simulates $F.TS.update$ procedure. I responds $SK_b = ((Z_{w^b|_1}, SN_{w^b|_1}), \dots, (Z_{w^b|_s}, SN_{w^b|_s}), (Z_{w^b}, SN_{w^b}))$ to F , where $w^b = w_1^* \dots w_s^*$ and $(Z_{w^b|_i}, SN_{w^b|_i}) = NULL$ if the last bit of $w^b|_k$ is 1.

When F finishes the over-threshold phase, she comes to the forge phase. At that time, F wants to forge a signature for M in period j , where $1 \leq j \leq b-1$. Let $w^j = w_1 \dots w_n$ denote the node corresponding to period j . F needs to query H_2 oracle to get $H_2(w^j \parallel M, U = rP, Z_{w^j})$ at first. If F can forge a valid signature $\langle j, \sigma = (U = rP, Z_{w^j}, FS) \rangle$, then

$$\begin{aligned} FS &= \alpha I_\epsilon + \sum_{m=1}^n \rho_{w^j|_m} h_{w^j|_m} I_\epsilon + r H_2(w^j \parallel M, U, Z_{w^j}) \\ &\quad + H_4(U, w^j, Z_{w^j}) \cdot I_\epsilon \end{aligned}$$

Since F has queried $H_2(w^j \| M, U, Z_{w^j})$, I can find $\langle w^j \| M, U = rP, Z_{w^j}, h = \gamma P, \gamma, * \rangle$ in H_2 table. Therefore, I can compute:

$$\begin{aligned} & \alpha\beta P \\ &= \alpha I_\varepsilon \\ &= FS - \sum_{m=1}^n \rho_{w^j|_m} h_{w^j|_m} I_\varepsilon - r\gamma P - H_4(U, w^j, Z_{w^j}) \cdot I_\varepsilon \\ &= FS - \sum_{m=1}^n \rho_{w^j|_m} h_{w^j|_m} I_\varepsilon - H_4(U, w^j, Z_{w^j}) \cdot I_\varepsilon - \gamma U \end{aligned}$$

where $\rho_{w^j|_m}$, $h_{w^j|_m}$ have been computed during simulating $FTS.update$ procedure and γ can be found in the tuple $\langle w^j \| M, U = rP, Z_{w^j}, h = \gamma P, \gamma, * \rangle$. The construction of algorithm I has been completed.

Now, we analyze the following three events and compute the probability for I to succeed.

Event E_1 : When F queries the signature oracle, I aborts. There is $\Pr[E_1] \leq (q_{H_2} - 1) \cdot q_S / (q - 1)$.

In H_2 table I maintains, the number of queries generated not by signing algorithm is $q_{H_2} - q_S$. Therefore, when the k -th signature query happens, in the worst case, there are at most $q_{H_2} - q_S + k - 1$ of H_2 queries defined. The probability for I to abort the k -th ($k \in \{1, 2, \dots, q_S\}$) signature query is at most $(q_{H_2} - q_S + k - 1) / (q - 1)$, where $q - 1$ is the size of the domain from which U (actually ϕ) is selected (that is the elements number of Z_q^*). Let ε_k denote the event that I aborts the k -th signature query. The following description is right:

$$\begin{aligned} \Pr[E_1] &= \Pr[\varepsilon_1 \cup \dots \cup \varepsilon_{q_S}] \leq \sum_{k=1}^{q_S} \Pr[\varepsilon_k] \\ &= \sum_{k=1}^{q_S} \frac{(q_{H_2} - q_S + k - 1)}{q - 1} \\ &= \frac{q_S(q_{H_2} - \frac{1}{2}q_S - \frac{1}{2})}{q - 1} \\ &\leq \frac{q_S(q_{H_2} - 1)}{q - 1} \end{aligned}$$

Event E_2 : F outputs d -over-threshold and the over-threshold phase is period b . There is $\Pr[E_2] = 1/T$.

Because F can't distinguish the simulation given by I from the real world, the probability that the period b which I guesses is equal to the period in which F enters her over-threshold phase is $1/T$.

Event E_3 : When I doesn't abort, F succeeds to forge a valid signature for a new message in period j , where $1 \leq j < b$. Obviously, there is $\Pr[E_3] \geq \varepsilon$.

Therefore, the probability for I to solve CDH problem is at least:

$$\begin{aligned} & \Pr[E_2] \cdot \Pr[\bar{E}_1] \cdot \Pr[E_3] \\ & \geq \frac{1}{T} (\varepsilon - \varepsilon \Pr[E_1]) \\ & \geq \frac{1}{T} (\varepsilon - \Pr[E_1]) \\ & \geq \frac{1}{T} (\varepsilon - \frac{q_S(q_{H_2} - 1)}{q - 1}) = \varepsilon' \end{aligned}$$

Below we analyze the total running time of I . Suppose that the bit operations in G_1 is at most $O(k^n)$. The running time of I is the running time of F plus the following time:

1. H_2 -query. One multiplication is needed to compute $h = \gamma P$ for a direct H_2 query. Three multiplications are needed to compute $h = \gamma P - (1/\phi) \cdot I_\varepsilon$ and $U = \phi R$ for an indirect H_2 query generated by signature query. Thus the time for H_2 -query is $O(k^n)$.
2. Signature query. We need to compute $FS = \sum_{m=1}^t \rho_{w^j|_m} h_{w^j|_m} I_\varepsilon + \phi\gamma R + H_4(U, w^j, Z_{w^j}) \cdot I_\varepsilon$ when signature oracle is queried. For each period $j < b$, no more than l multiplications are needed to compute $\sum_{m=1}^t \rho_{w^j|_m} h_{w^j|_m} I_\varepsilon$, so the time for all periods is no more than $O(T \cdot k^n)$. For $\phi\gamma R + H_4(U, w^j, Z_{w^j}) \cdot I_\varepsilon$, the time is no more than $O(k^n)$.
3. $FTS.update$ simulation. Six multiplications are needed at most for once key update. So the time for b times key update is no more than $O(T \cdot k^n)$.
4. Response to over-threshold query. Note that the time to compute $\sum_{m=1}^{s-1} \rho_{w^b|_m} h_{w^b|_m} I_\varepsilon$ and $\sum_{m=1}^{i-1} \rho_{w^b|_m} h_{w^b|_m} I_\varepsilon$ has been considered in signature query. The total time to compute $r_{w^b} I_\varepsilon$ and all $r_{w^b|_m} I_\varepsilon (1 \leq m \leq s)$ is at most $O(k^n)$.
5. To resolve CDH problem, the time is $O(k^n)$ to compute γU and $H_4(U, w^j, Z_{w^j}) \cdot I_\varepsilon$.
6. The simulation of $VIEW_F$ in $FTS.key$, $FTP.signature$,

$FTS.update$ algorithms. The total time is $O(T \cdot k^n)$.

Thus, the total running time of I is at most $t + O(T \cdot k^n) = t'$.

It is contractive to the assumption that the group G_1 generated by $IG(1^k)$ is a (t', ε') -break CDH group. Therefore, the theorem follows.

VI. CONCLUSIONS

Based on the structure of binary tree, we construct an efficient forward secure threshold signature scheme from bilinear pairings. All the running costs of key generation, key update, signing and verifying algorithms are independent of the total number of time periods T . Finally, we prove the proposed scheme is robust and forward secure when CDHP is hard.

ACKNOWLEDGMENT

This research is supported by Natural Science Foundation of China (60703089), the National High-Tech R & D Program (863 Program) of China (2006AA012110) and National Cryptologic Development Foundation of China.

REFERENCES

- [1] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612-613, 1979.
- [2] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Robust threshold DSS signatures," *Advances in Cryptology-Eurocrypt'96*, 1996, pp. 354-371.
- [3] A. Herzberg, M. Jakobsson, S. Jarecki, and M. Yung, "Proactive public key and signature systems," In *Proc of the 4th Annual Conference on Computers and Communication Security*, 1997, pp. 100-110.
- [4] R. Anderson, "Two remarks on public key cryptology," *Invited Lecture, 4th ACM Conference on Computer and Communications Security*, 1997.
- [5] M. Bellare and S. Miner, "A forward-secure digital signature scheme," *Advances in Cryptology-CRYPTO'99*, 1999, pp. 431-448.
- [6] M. Abdalla and L. Reyzin, "A new forward-secure digital signature scheme," *Advances in Cryptology-Asiacrypt'00*, 2000, pp. 116-129.
- [7] G. Itkis and L. Reyzin, "Forward-secure signatures with optimal signing and verifying," *Advances in Cryptology-CRYPTO'01*, 2001, pp. 499-514.
- [8] A. Kozlov, and L. Reyzin, "Forward-secure signatures with fast key update," *Security in communication Networks*, 2002, pp. 247-262.
- [9] F. Hu, C. H. Wu and J. D. Irwin, "A new forward secure signature scheme using bilinear maps," *Cryptology ePrint Archive, Report 2003/188*, 2003.
- [10] B. G. Kang, J. H. Park, and S. G. Halm, "A new forward secure signature scheme," *Cryptology ePrint Archive, Report 2004/183*, 2004.
- [11] J. Yu, F. Y. Kong, and D. X. Li, "An Efficient Forward Secure Signature Scheme," *Journal of Shanghai Jiaotong University (Science)*, Vol. E-11, No. 2, pp. 242-247, 2006.
- [12] J. Camenisch, and M. Koprowski, "Fine-grained forward-secure signature schemes without random oracles," *Discrete Applied Mathematics*, vol. 154, no. 2, pp. 175-188, 2006.
- [13] X. Boyen, H. Shacham, E. Shen, and B. Waters, "Forward Secure Signatures with Untrusted Update," *Proceedings of the 13th ACM conference on Computer and communications security*, 2006, pp. 191-200.
- [14] M. Abdalla, S. Miner, and C. Namprempre, "Forward-secure threshold signature schemes," *Topics in Cryptology-CT-RSA'01*, 2001, pp. 441-456.
- [15] Z. J. Tzeng, and W. G. Tzeng, "Robust forward signature schemes with proactive security," In *Proc. PKC 2001*. LNCS 1992, Berlin: Springer-Verlag, 2001. 264~276.
- [16] H. Wang, G. Qiu, D. Feng, and G. Xiao, "Cryptanalysis of Tzeng-Tzeng Forward-Secure Signature Schemes," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E89-A, no. 3, pp. 822-825, 2006.
- [17] C. K. Chu, L. S. Liu, and W. G. Tzeng, "A threshold GQ signature scheme," *Cryptology ePrint Archive, Report 2003/016*, 2002.
- [18] J. Yu, F. Y. Kong, and R. Hao, "Forward Secure Threshold Signature Scheme from Bilinear Pairings," In the *Second International Conference on Computational Intelligence and Security*, 2007, pp. 587-597.
- [19] M. Bellare, and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," In *First ACM Conference on Computer and Communications Security*, 1993, pp. 62-73.
- [20] R. Gennaro, S. Jarecki, H. Krawczyk and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystems," *Advances in Cryptology-Eurocrypt'99*, 1999, pp. 295-310.
- [21] C. Gentry, and A. Silverberg, "Hierarchical ID-based cryptography," *Advances in Cryptology-ASIACRYPT'02*, 2002, pp. 548-566.

Jia Yu was born in China in 1976. He received the BS, MS, and PhD degrees in computer science from Shandong University, Shandong, China, in 2000, 2003, and 2006, respectively.

He became a lecturer, an associate professor of computer science in the College of Information Engineering at Qingdao University, China, in 2006 and 2007, respectively. He is currently an associate professor in the College of Information Engineering at Qingdao University, China. His research interests include encryption, digital signature, cryptographic protocol and network security.

Dr. Yu currently is a member of Chinese Association for cryptologic Research and Chinese Computer Federation.

Fanyu Kong was born in China in 1978. He received the BS, MS, and PhD degrees in computer science from Shandong University, Shandong, China, in 2000, 2003, and 2006, respectively.

He became a lecturer of computer science in the institute of Network Security at Shandong University, China, in 2006. He is currently a fellow in the institute of Network Security at Shandong University, China. His research interests include cryptography and network security.

Dr. Kong currently is a member of Chinese Association for cryptologic Research.

Xiangguo Cheng was born in China in 1969. He received the BS, MS, and PhD degrees in Applied Math from Jilin University, Tongji university, and Xiandianzi Univeristy, China, in 1992, 2003, and 2006, respectively.

He became an associate professor of computer science in the College of Information Engineering at Qingdao University, China, in 2007. He is currently an associate professor in the College of Information Engineering at Qingdao University, China. His research interests include digital signature, cryptographic protocol and network security.

Dr. Cheng currently is a member of Chinese Association for cryptologic Research.