Forward Secure Threshold Signature Scheme from Bilinear Pairings

Jia Yu[1], Fanyu Kong[2], and Rong Hao[1,2]

[1] College of Information Engineering, Qingdao University,
266071 Qingdao, China
{yujia,hr}@qdu.edu.cn
[2] Institute of Network Security, Shandong University,
250100 Jinan, China
phd_kong@yahoo.com

Abstract. A forward secure threshold signature scheme from bilinear pairings is presented in this paper. Compared with previous forward secure threshold signature schemes against malicious adversary, this scheme needs very few interactions and is very efficient. A new interactive zero-knowledge proof protocol is presented and its non-interactive version can verify the validity of part signatures in this scheme. At last, we prove that the scheme is robust and forward secure in the random oracle model.

1 Introduction

Exposure of secret keys is one of the greatest threats to the security of a digital signature. Therefore, how to deal with the problem of secret key exposure in signatures is very important. Threshold signature is used to void secret key exposure. In a (t+1, n) threshold signature[1, 2], a secret key is distributed into n servers, and each server has a share of the secret key. Only more than t servers can jointly generate signatures. In comparison, forward secure signature[3~5] can reduce the damage of secret key exposure. In this paradigm, the whole lifetime of signature is divided into several time periods, the secret key is updated in each period by a one-way function, at the same time, the old key is destroyed. As a result, even if the current secret key is exposed, the adversary can't forge signatures for past time periods. Forward secure threshold signature combines the merits of the both kinds of signatures, as a result, it satisfies that if an adversary breaks into no more than t servers, she can't forge any signature; even if an adversary breaks into more than t servers, she can't forge signatures pertaining to previous time periods.

Related works. Abdalla *et al.* [6] firstly present a forward secure threshold signature scheme against malicious adversary based on the scheme in [3]. However, in their scheme, the size of both the public key and the secret key is very large, what's more, the scheme needs a lot of interactions because distributed multiplication of many values protocol is used. Distributed multiplication of many values protocol is very inefficient, and it is still an open problem to improve its efficiency [6]. Following by

Y. Wang, Y. Cheung, and H. Liu (Eds.): CIS 2006, LNAI 4456, pp. 587–597, 2007.

Abdalla's work, another forward secure threshold signature with proactive property [7] is suggested, which needs shorter keys, however, has lower efficiency. Wang *et al.* [8] point out the scheme in [7] is insecure. Chu *et al.* present a forward secure threshold signature scheme with weak security as an extension of his main work [9], but it can't tolerate malicious adversary and has not any proofs of forward security.

Our contribution. Based on [5], a forward secure threshold signature scheme *FTS* from bilinear pairings is presented in this paper. We name it as scheme *FTS*.

In addition, we present a new interactive zero-knowledge proof protocol named as *Proof-VS*, and prove that it is complete, sound and zero-knowledge. Then we convert *Proof-VS* into a non-interactive version *NIProof-VS* which is used to verify the validity of part signatures in our scheme by using a collision-resistant hash function. With necessary changes, *NIProof-VS* can be used in other threshold schemes from pairings effectively, too. Note that scheme *FTS* can void using distributed multiplication of many values thanks to using bilinear pairings.

Scheme *FTS* is very efficient. There are only once interaction in update algorithm and twice interactions in signing algorithm in this scheme. The running time of both the key generation and the key update algorithm is independent of the total number of time periods T in the scheme. And the signing and verifying costs are only logarithmic in T. The new scheme is robust against malicious adversary. Finally, we prove it is forward secure in the random oracle model assuming CDH problem is hard.

Organization. In section 2, we introduce the preliminaries of our work, including communication model, definition and related mathematical background. The presented zero-knowledge proof protocol and the concrete description of the scheme are given in Section 3. We compare the efficiency of our scheme with related schemes and provide the security proof of our scheme in section 4 and 5, respectively. In section 6, further discussion is given. Finally, we conclude this paper in the last Section.

2 Preliminaries

2.1 Model and Definition

(1) Communication Model

There exists a trusted dealer during the key generation phase. We assume that the participants in our scheme include a set of n players $\{1,2,\ldots,n\}$ who are connected by a broadcast channel. Additionally, they can securely communicate over private point-to-point channels. Furthermore, the system works in a synchronous communication model; that is, all participating players have a common clock and, thus, can send their messages simultaneously in a particular round of a protocol. Finally, the whole lifetime of signature is divided into T time periods. At the end of each time period, the participants update their shares all together.

(2) Forward Secure Threshold Signature Scheme

Definition 1 (Key-evolving threshold signature scheme). A key-evolving threshold signature scheme is a quadruple of algorithms, $FTS(t,s,n)=(FTS.GEN, FTS.UPD, FTS.SIG, FTS.VER)$, where, t is the maximum number of corrupted players; s is the

minimum number of honest players so that signature computation is feasible; n is the total number of players,

FTS.GEN: 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 of secret key for all players.

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

FTS.SIG: the signing algorithm, inputs the current time period j and a message M, and the participant players jointly generate a signature <j, tag> of M for period j.

FTS.VER: the verification algorithm, inputs the public key PK, a message M and a signature <j, tag>, and returns 1 if <j, tag> is a valid signature of M or 0, otherwise.

$SK_j^{(i)}$ is a share player i holds in period j. Assume that $SK_j^{(i)}$ always contains the value j and the total number of time periods T. If <j, tag> is a valid signature generated in *FTS.SIG* algorithm, then *FTS.VER*(M, <j, tag>)=1.

The adversary model in ROM. The adversary F chooses players to corrupt at the beginning of the game. She runs in three stages: in the chosen-message attack (cma) phase, F has access to the signing oracle, and can query to obtain the signature of any message she selects under the current secret key. At the end of each time period, the adversary can choose whether to stay in the same phase or switch to the over-threshold phase. In the over-threshold phase, for a particular time period b, the adversary may corrupt a set of players of size t+1 or greater. It means F can learn the secret key. In the forgery phase, the adversary outputs her forgery, i.e. a signature message pair. We consider an adversary successful if she forges a signature of some new message for some period prior to the time period b. The adversary is allowed to query a random oracle H corresponding to a collision-resistant hash function.

Definition 2 (Forward secure threshold signature scheme). A key-evolving threshold signature scheme is a forward secure threshold signature scheme if there is no such an adversary described above that can forge a signature <j, tag> for some new message M *s.t. FTS.VER*(M, <j, tag>)=1 and $j<b$.

2.2 Cryptographic Assumptions

Let G_1 and G_2 be two groups of some prime order q, where G_1 is represented additively and G_2 is represented multiplicatively. And let $P \in G_1$ be a generator of G_1. A bilinear pairing is an efficiently computable map $e: G_1 \times G_1 \to G_2$, which satisfies:

1. Bilinear: For all $P, Q \in G_1$ and $a, b \in Z$, there is $e(aP, bQ) = e(P, Q)^{ab}$.
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 $e(P,Q)$ for any $P, Q \in G_1$.

Decision Diffie-Hellman (DDH) problem: Given (P, aP, bP, cP), where $a, b, c \in Z_q$, decide whether $c = ab$ in Z_q. Computation Diffie-Hellman (CDH) problem: Given (P, aP, bP), where $a, b \in Z_q$, compute abP.

Definition 3 (GDH group). A prime order group G is a GDH group if DDH problem can be solved in polynomial time but no probabilistic algorithm solves CDH problem.

The Weil and Tate pairings are practical example of the bilinear pairing. Using the Weil or Tate pairing, certain elliptic curves can be used as GDH groups.

IG is a GDH parameter generator if only it takes security parameter k, outputs two groups G_1 and G_2 and an admissible pairing $e: G_1 \times G_1 \to G_2$.

3 The Forward Secure Threshold Signature Scheme

3.1 Building Blocks

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.

(1) The zero-knowledge proof protocols we present

Prover P wants to convince verifier V that she knows the values $b_i (i = 1...n)$ that satisfy $G_i = b_i P_0 (i = 1...n), H' = \sum_{i=1}^{n} b_i P_i$.

Firstly, we give an interactive protocol $\Pr oof - VS(P_0; P_1, ..., P_n; G_1, ..., G_n; H')$:

①P selects $w_i \in_R Z_q (i = 1...n)$, computes $E_i = w_i P_0 (i = 1...n)$ and $F = \sum_{i=1}^{n} w_i P_i$, and sends these values to V.

②V randomly chooses $c \in_R Z_q$, and sends it to P.

③P computes $r_i = w_i - b_i c, (i = 1...n)$, and sends them to V.

④V verifies: $E_i \stackrel{?}{=} r_i P_0 + c G_i (i = 1...n)$ and $F \stackrel{?}{=} cH' + \sum_{i=1}^{n} r_i P_i$. If equations are right, V believes P; otherwise, doesn't.

Theorem 1. *Proof-VS* is complete, sound and zero-knowledge.

Proof. (Sketch)

(1)Completeness : If P and V are honest and P knows b_i, then the following both equations are right:

$$E_i = w_i P_0 = (r_i + b_i c) P_0 = r_i P_0 + c G_i, (i = 1...n),$$

$$F = \sum_{i=1}^{n} w_i P_i = \sum_{i=1}^{n} (r_i + b_i c) P_i = cH' + \sum_{i=1}^{n} r_i P_i .$$

So V believes P.

(2)Soundness: If P doesn't know some b_i, after P sends E_i and F to V, and then gets c from V, P is only able to guess r_i. The probability that P makes V believe him is $1/q$ at most. Therefore the soundness can be satisfied.

(3)Zero-knowledge: We can construct a simulator S to simulate the view of any verifier. S selects $c' \in_R Z_q$ and $r_i' \in_R Z_q (i = 1...n)$. She computes $E_i' = r_i' P_0 + c' G_i$,

$(i=1...n)$ and $F'=c'H'+\sum_{i=1}^{n} r_i'P_i$. The distributions of E_i', F', c_i' and r' are statistically indistinguishable from those of E_i, F, c, r_i in the real view.

Then, convert *Proof-VS* into a non-interactive version by a collision-resistant hash function: $H: G \to Z_q$. The protocol $NI\,\mathrm{Pr}oof-VS(P_0;P_1,...,P_n;G_1,...,G_n;H')$ is described as follows:

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

②V verifies:

$$\overset{?}{c} = H(P_0 \| P_1 \| ... \| P_n \| G_1 \| ... \| G_n \| H' \| r_1 P_0 + cG_1 \| ... \| r_n P_0 + cG_n \| cH' + \sum_{i=1}^{n} r_i P_i)$$

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

(2) Distributed random secret generation (*Joint-RVSS*) protocol

All players jointly and verifiably generate a random secret ρ and each player *i* has a share ρ_i of the secret. Any *t*+1 players can reconstruct the secret ρ, however, *t* players can't get any useful message about it. The public commits include ρP_0 and $\rho_i P_0 (i=1...n)$. We use the *Joint-Exp-RSS* protocol in [10] as the *Joint-RVSS* protocol in our scheme. So we will use the security results of it to prove the security of our threshold signature scheme. As is proved in [10], the view of an adversary corrupting *t* players can be simulated when the commit ρP_0 is taken as input. Here we skip the detailed descriptions of this protocol and its security proof.

3.2 Notations

Our scheme uses a binary tree structure similar to that in [5] which is a variant of the tree structure used in the HIBS scheme in [11]. If we use a full binary tree with depth l, then the number of time periods is $T=2^{l+1}-1$ (labeled 0 through *T*-1). 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 $w^{j+1}=w'1$, where w' is the longest string such that $w'0$ 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 $w'0$ is a prefix of w^j, $SK_j^{(i)}$ contains the share $S_{w'1}^{(i)}$ of secret key of node $w'1$. 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 $S_{w^j0}^{(i)}$ and $S_{w^j1}^{(i)}$ of w^j0 and w^j1, respectively. And then push $S_{w^j1}^{(i)}$ and $S_{w^j0}^{(i)}$ onto the stack orderly. The new top is $S_{w^j0}^{(i)}$ and indeed $w^{j+1}=w^j0$. 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 $S_{w^{j+1}}^{(i)}$.

3.3 Description of Our Scheme

Let IG be a GDH parameter generator for which the GDH assumption holds. Our forward security threshold signature scheme is constructed as follows:

(1) *FTS.GEN* : Input a security parameter *k*, the total number of time periods $T=2^{l+1}-1$ (l is the depth of the binary tree).

①Run IG (1^k) to generate groups G_1 and G_2 of some prime order q and an admissible pairing $e: G_1 \times G_1 \to G_2$.

②Select a random generator $P \in G_1$ and a random secret $\alpha \in Z_q$, and set $Q=\alpha P$. Choose $a_i \in_R Z_q (i=1..t)$, set $f(x)=\alpha+\sum_{i=1}^{t} a_i x^i \pmod q$, and then compute $\alpha_i = f(i), (i=1...n)$.

③Choose cryptographic hash functions $H: G_1 \to Z_q$, $H_1: \{0,1\}^* \to G_1$, $H_2: \{0,1\}^* \times \{0,1\}^* \times G_1 \to G_1$.

④The public key is $PK=(G_1, G_2, e, P, Q, l, H, H_1, H_2)$. Compute and broadcast $R_\varepsilon^{(i)}=\alpha_i P (i=1...n)$. Send share α_i to player $i (i=1...n)$ secretly. Each player $i (i=1...n)$ computes $SN_\varepsilon^{(i)}=\alpha_i H_1(\varepsilon)$, and then pushes $S_\varepsilon^{(i)}=(SN_\varepsilon^{(i)})$ onto stack $ST^{(i)}$.

(2) *FTS.UPD*: Input the current time period k. Let $w=w_1...w_n$ denote the node corresponding to k. Firstly each player $i (i=1...n)$ pops the node secret share $S_w^{(i)}$ off the stack $ST^{(i)}=SK_k^{(i)}$, and then does as follows:

①If w is an internal node, parses $S_w^{(i)}=(R_{w|_1}, R_{w|_2}, ..., R_{w|_{n-1}}, R_w, SN_w^{(i)})$. By executing twice $Joint-RVSS$ simultaneously, all players jointly generate two random values $\rho_{w0}, \rho_{w1} \in Z_q$. Player i gets shares $\rho_{w0}^{(i)}, \rho_{w1}^{(i)} \in Z_q$ and public commits $R_{w0}=\rho_{w0}P$, $R_{w1}=\rho_{w1}P$, $R_{w0}^{(j)}=\rho_{w0}^{(j)}P$, $R_{w1}^{(j)}=\rho_{w1}^{(j)}P$, where $j=1,...,n$. Player i then computes $SN_{w0}^{(i)}=SN_w^{(i)}+\rho_{w0}^{(i)}H_1(w0)$, and $SN_{w1}^{(i)}=SN_w^{(i)}+\rho_{w1}^{(i)}H_1(w1)$. She erases $S_w^{(i)}$ and pushes $S_{w1}^{(i)}=(R_{w|_1}, R_{w|_2}, ..., R_{w|_{n-1}}, R_w, R_{w1}, SN_{w1}^{(i)})$ and $S_{w0}^{(i)}=(R_{w|_1}, R_{w|_2}, ..., R_{w|_{n-1}}, R_w, R_{w0}, SN_{w0}^{(i)})$ onto the stack $ST^{(i)}$ orderly at last.

②If w is a leaf, then only erases $S_w^{(i)}$.

(3) *FTS.SIG* : Input the time period k and a message M. Let $w = w_1 ... w_n$ denote the node corresponding to period k. Firstly each participant player i reads the node secret share $S_w^{(i)}$ from the top of the stack $ST^{(i)} = SK_k^{(i)}$, and then does as follows:

①Parses $S_w^{(i)} = (R_{w|_1}, R_{w|_2}, ..., R_{w|_{n-1}}, R_w, SN_w^{(i)})$. All players jointly generate a random secret $r \in Z_q$ by executing $Joint-RVSS$. Player i gets the share $r^{(i)} \in Z_q$ and the public commits $U = rP$, $U^{(j)} = r^{(j)} P$, where $j = 1, ..., n$.

②Then player i computes $P_M = H_2(M, k, U)$, $FS^{(i)} = SN_w^{(i)} + r^{(i)} P_M$, and executes $NIProof-VS(P; H_1(\varepsilon), H_1(w|_1), ..., H_1(w|_n), P_M; R_\varepsilon^{(i)}, R_{w|_1}^{(i)}, ..., R_{w|_n}^{(i)}, U^{(i)}; FS^{(i)})$ in order to prove the part signature $FS^{(i)}$ which she provides satisfies that $FS^{(i)} = \alpha_i H_1(\varepsilon) + \sum_{m=1}^{n} \rho_{w|_m}^{(i)} H_1(w|_m) + r^{(i)} P_M$, and these α_i, $\rho_{w|_m}^{(i)}, (m = 1...n)$ and $r^{(i)}$ satisfy $R_\varepsilon^{(i)} = \alpha_i P$, $R_{w|_m}^{(i)} = \rho_{w|_m}^{(i)} P, (m = 1...n)$, $U^{(i)} = r^{(i)} P$. If the verification passes, it means the participant player i provides a valid part signature.

③Any set B of t+1 players who pass the verification of *NIProof-VS*, then compute $FS = \sum_{i \in B} C_{Bi} FS^{(i)}$ and publish signature $<k, \sigma = (U, FS)>$ and $R_{w|_m} (m = 1, ..., n)$.

(4) *FTS.VER* : Input a message M and a signature $<k, \sigma = (U, FS)>$ and $R_{w|_m} (m = 1, ..., n)$. Let $P_M = H_2(M, k, U)$. Return 1 if

$$e(P, FS) = e(Q, H_1(\varepsilon)) \cdot \prod_{m=1}^{n} e(R_{w|_m}, H_1(w|_m)) \cdot e(U, P_M)$$

or 0, otherwise.

4 Efficiency Comparisons

The complexity analysis is considered in terms of T according to the method in [5]. Therefore, the complexity of all computations independent of T is $O(1)$. l' is a security parameter in [6, 7].

Compare the efficiency of our scheme with the schemes against malicious adversary in [6, 7] in Table 1. The running time of both key generation and key update algorithms in our scheme is independent of T, so the complexity is $O(1)$, as opposed to $O(1)T$ in [6] and $O(1)l'T$ in [7]. Signing and verifying in our scheme can all be finished in $O(1)\log T$ time. So their costs are only linear in $\log T$. The efficiency of verifying algorithm, to some extent, depends on the efficiency of pairing computation. With a good pairing computation algorithm, we can have an efficient verifying algorithm.

The total interactions in our scheme are the fewest in the three schemes, too. There is no interaction in our key generation algorithm. Key update algorithm will execute twice *Joint-RVSS* simultaneously, but only needs once interaction. In signing algorithm twice interactions are needed in total, one happens in *Joint-RVSS* and the other happens in *NIProof-VS*.

Table 1. Efficiency comparisons

	The scheme in [6]	The scheme in [7]	Our scheme
FTS.GEN time and interactions	$O(1)T$	$O(1)l'T$	$O(1)$
	0	1	0
FTS.UPD time and interactions	$O(1)T$	$O(1)l'T$	$O(1)$
	1	2	1
FTS.SIG time and interactions	$O(1)T$	$O(1)l'T$	$O(1)\log T$
	$2l'$	2	2
FTS.VER time	$O(1)T$	$O(1)l'T$	$O(1)\log T$

5 Security Analysis

Theorem 2. Let $PK = (G_1, G_2, e, P, Q, l, H, H_1, H_2)$ and $SK_0^{(i)} = S_\varepsilon^{(i)} (i = 1...n)$ be the public key and the shares generated by *FTS.GEN*, respectively; Let the shares of secret key be updated by *FTS.UPD*; Let $< k, \sigma = (U, FS) >$ and $R_{w|_m} (m = 1,...,n)$ be a signature generated by *FTS.SIG* on input a message M for period k. Then *FTS.VER*(M, <k,(U,FS)>=1.

Proof

$$
\begin{aligned}
e(P, FS) &= e(Q, H_1(\varepsilon)) \cdot \prod_{m=1}^{n} e(R_{w|_m}, H_1(w|_m)) \cdot e(U, P_M) \\
&= e(\alpha P, H_1(\varepsilon)) \cdot \prod_{m=1}^{n} e(\rho_{w|_m} P, H_1(w|_m)) \cdot e(rP, P_M) \\
&= e(P, \alpha H_1(\varepsilon)) \cdot \prod_{m=1}^{n} e(P, \rho_{w|_m} H_1(w|_m)) \cdot e(P, rP_M) \\
&= e(P, \alpha H_1(\varepsilon) + \sum_{m=1}^{n} \rho_{w|_m} H_1(w|_m) + rP_M) \\
&= e(P, \sum_{i \in B} (C_{Bi} \alpha_i H_1(\varepsilon) + \sum_{m=1}^{n} C_{Bi} \rho_{w|_m}^{(i)} H_1(w|_m) + C_{Bi} r^{(i)} P_M)) \\
&= e(P, \sum_{i \in B} C_{Bi} FS^{(i)}) \\
&= e(P, FS)
\end{aligned}
$$

Theorem 3. The *FTS* scheme is a key-evolving (t,s,n) -threshold signature scheme against malicious adversary for $s \geq t+1$, $n \geq 2t+1$.

Proof. (Sketch) It is because *NIProof-VS* and *Joint-RVSS* protocols can detect the dishonest players in the protocol and s honest players can make *FTS.UPD* and *FTS.SIG* algorithms be executed properly for $s \geq t+1$, $n \geq 2t+1$. According to theorem 2, the scheme can tolerant a malicious adversary corrupting t players.

Theorem 4. The *FTS*(t,s,n) scheme is a forward secure threshold signature scheme in the random oracle model in the presence of malicious adversary for $s \geq t+1$, $n \geq 2t+1$.

Proof. (Sketch) The security of our threshold scheme is based on CDH assumption. Because *FS.PKE* [5] is forward secure, assuming that CDH problem is hard, we only need prove our scheme is forward secure as long as *FS.PKE* is forward secure.

Let F be an adversary working against the security of FTS. F runs in three stages: the chosen-message attack phase, cma; the over-threshold phase, overthreshold; and the forgery phase, forge. We want to construct an algorithm I against the security of $FS.PKE$ using F as a subroutine. I works in three stages: the chosen-message attack phase, cma; the break-in phase, breakin; and the forgery phase, forge. I has access to both a singing oracle SIG' and a hashing oracle H_2'.

In the cma phase, F is allowed to make queries to a signing oracle $FTS.SIG$ and a hash oracle H_2. Therefore, we need to simulate both of these oracles. 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 $1...t$.

The simulation of $VIEW_F$ in $FTS.GEN$: Because $f(x)$ is a random polynomial in Z_q, α_i is a random value in Z_q. That is, $SN_\varepsilon^{(i)}$ is distributed uniformly in G_1. We can pick values for $\alpha_i (i=1...t)$ at random from Z_q. And then compute $SN_\varepsilon^{(i)}$, $R_\varepsilon^{(i)}$ $(i=1...t)$. For each $R_\varepsilon^{(j)} (j=t+1...n)$, compute $R_\varepsilon^{(j)} = \alpha_j P = (\lambda_{j,0} \cdot \alpha + \sum_{i=1}^{t} \lambda_{j,i} \cdot \alpha_i) P$ $= \lambda_{j,0} Q + \sum_{i=1}^{t} \lambda_{j,i} \cdot R_\varepsilon^{(i)}$, where $\lambda_{j,i}$ is computable Lagrange interpolation coefficient.

The simulation of $VIEW_F$ in $FTS.UPD$: Because the shares of secrets ρ_{w0}, ρ_{w1} are distributed uniformly in Z_q, we can pick random values $\rho_{w0}^{(i)}, \rho_{w1}^{(i)} (i=1...t)$ in Z_q for F. It is easy to compute $SN_{w1}^{(i)}, SN_{w0}^{(i)}$ and provide them to F. According to the security proof of $Joint-RVSS$, taking as input R_{w0}, R_{w1}, we can simulate the $Joint-RVSS$ protocol to get $VIEW_F$ including $R_{w0}^{(i)}, R_{w1}^{(i)} (i=1...n)$ in this protocol.

The simulation of signing oracle $FTS.SIG$: For each query <M, k> made by F, query SIG' oracle and get signature <k, (U, FS)>. Then return <k, (U, FS)> to F as the answer to her signing query.

The simulation of H_2 hash oracle: For each query (M, k, U) made by F, query H_2' oracle, and then give the answer to F directly.

The simulation of $VIEW_F$ in $FTS.SIG$: For a message <M, k>, take as input U got from query $FTS.SIG$ oracle to simulate the $Joint-RVSS$ protocol, therefore, we can get $VIEW_F$ including $U^{(i)} = r^{(i)} P$ $(i=1...n)$ in the protocol. $H_2(M,k,U)$ can be got by querying H_2 hash oracle and $FS_i (i=1...t)$ can be computed according to $r_i (i=1...t)$. $SN_w^{(i)}$ and $R_w^{(i)}$ $(i=1...t)$ can be got from simulation of $FTS.UPD$. With FS obtained from the query of the signing oracle $FTS.SIG$, we can compute $FS_i (i=t+1...n)$ which F views by the same means of Lagrange interpolation in simulation of $FTS.UPD$. Simulate the $NIProof-VS$ protocol at last.

In the break-in stage: When F has finished cma stage and decides to switch to over-threshold stage b, we run I in break-in stage. So we can get current secret key SK_b and provide it to F.

In forge stage: When F finishes her break-in stage, she can construct a forged signature $< k, (U, FS) >$ and $R_{w|_m} (m=1,...,n)$ for $< M, k >, (k < b)$ to I. Finally, I outputs $< k, (U, FS) >$ and $R_{w|_m} (m=1,...,n)$ as her forgery.

It means that there must be an algorithm *I* against the forward security of *FS.PKE* if there exists an adversary *F* working against the forward security of our *FTS* scheme. When CDH problem is hard, *FS.PKE* is forward secure according to the security theorem 1 in [5]. In another word, there is no PPT algorithm against the forward security of *FS.PKE*. So there is no PPT adversary against the forward security of the *FTS* scheme. According to definition 2 and theorem 3, the *FTS*(*t,s,n*) scheme is a forward secure threshold signature scheme in the random oracle model in the presence of malicious adversary for $s \geq t+1$, $n \geq 2t+1$.

6 Further Discussion

(1) Proactive security
We can add proactive security to this scheme. Proactive secret sharing has been presented in [12]. In the paradigm, shares are periodically renewed without change the secret key. Therefore, the shares got by the adversary in one period are useless after they are renewed. Nikov and Nikova [13] point out scheme [12] is insecure against a mobile adversary. We can apply similar method to this scheme to get proactive security against static adversary. Because of limited space, the concrete content is neglected here.

(2) Storage space
In this scheme, the secret share size and the signature size are not independent to the total time periods *T*, but have log*T* complexities. Fortunately, our scheme is based on the bilinear parings that are constructed from certain elliptic curve. Thus the scheme works on a small finite field and has smaller storage space than other schemes as long as the total number of time periods is not very large.

7 Conclusion

A forward secure threshold signature scheme from bilinear pairings is given in this paper, which is the extended version of [14]. The scheme is very efficient and needs very few interactions. As an additional result, we present an interactive zero-knowledge proof protocol and convert it into a non-interactive protocol in order to verify the validity of part signatures in the scheme. The scheme is robust and forward secure assuming CDH problem is hard.

Acknowledgement

We would like to thank anonymous referees of 2006 International Conference on Computational Intelligence and Security (CIS'06) for the suggestions to improve this paper. This work is supported by the National High-Tech R & D Program (863 Program) of China.

References

1. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) Advances in Cryptology - CRYPTO '89. LNCS, vol. 435, pp. 307–315. Springer, Heidelberg (1990)
2. Shoup, V.: Practical threshold signature. In: Preneel, B. (ed.) Advances in Cryptology - EUROCRYPT 2000. LNCS, vol. 1807, pp. 207–220. Springer, Heidelberg (2000)
3. Bellare, M., Miner, S.: A forward-secure digital signature scheme. In: Wiener, M.J. (ed.) Advances in Cryptology - CRYPTO '99. LNCS, vol. 1666, pp. 431–448. Springer, Heidelberg (1999)
4. Itkis, G., Reyzin, L.: Forward-secure signatures with optimal signing and verifying. In: Kilian, J. (ed.) Advances in Cryptology - CRYPTO 2001. LNCS, vol. 2139, pp. 499–514. Springer, Heidelberg (2001)
5. Kang, B.G., Park, J.H., Halm, S.G.: A new forward secure signature scheme. Cryptology ePrint Archive, Report 2004/183 (2004)
6. Abdalla, M., Miner, S., Namprempre, C.: Forward-secure threshold signature schemes. In: Naccache, D. (ed.) Topics in Cryptology - CT-RSA 2001. LNCS, vol. 2020, pp. 441–456. Springer, Heidelberg (2001)
7. Tzeng, Z.J., Tzeng, W.G.: Robust forward signature schemes with proactive security. In: Kim, K.-c. (ed.) Public Key Cryptography. LNCS, vol. 1992, pp. 264–276. Springer, Heidelberg (2001)
8. Wang, H., Qiu, G., Feng, D., Xiao, G.: Cryptanalysis of Tzeng-Tzeng Forward-Secure Signature Schemes. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E89-A(3), 822–825 (2006)
9. Cheng-Kang Chu, Li-Shan Liu, Wen-Guey Tzeng. A threshold GQ signature scheme. Cryptology ePrint Archive, Report 2003/016 (2002)
10. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems, Advances in Cryptology-Eurocrypt'99. In: Stern, J. (ed.) Advances in Cryptology - EUROCRYPT '99. LNCS, vol. 1592, pp. 295–310. Springer, Heidelberg (1999)
11. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography, Advances in Cryptology-Asiacrypt 2002. In: Zheng, Y. (ed.) Advances in Cryptology - ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
12. Herzberg, A., Jarecki, S., Krawczyk, H., Yung, M.: Proactive Secret Sharing, or: how to cope with perpetual leakage, Advances in Crytptolgy-Crypto'95. In: Coppersmith, D. (ed.) Advances in Cryptology - CRYPTO '95. LNCS, vol. 963, pp. 339–352. Springer, Heidelberg (1995)
13. Nikov, V., Nikova, S.: On proactive secret sharing schemes. In: Handschuh, H., Hasan, M.A. (eds.) Selected Areas in Cryptography. LNCS, vol. 3357, pp. 314–331. Springer, Heidelberg (2004)
14. Yu, J., Kong, F., Hao, R.: A New Forward Secure Threshold Signature Scheme. In: International Conference on Computational Intelligence and Security 2006, pp. 1243–1246. IEEE Press, New York (2006)