Robust Forward-Secure Signature Schemes with Proactive Security*
(Extended Abstract)

Wen-Guey Tzeng and Zhi-Jia Tzeng

Department of Computer and Information Science
National Chiao Tung University
Hsinchu, Taiwan 30050
{tzeng,zjtzeng}@cis.nctu.edu.tw

Abstract. The secret key of a forward-secure signature scheme evolves at regular intervals, but the public key is fixed during the lifetime of the system. This paper enhances the security of Abdalla and Reyzin's forward-secure signature scheme via threshold and proactive mechanisms. In our threshold forward-secure signature scheme, we combine multiplicative and polynomial secret sharing schemes to form a threshold forward-secure signature scheme. We develop a special proof system to prove robustness of our scheme.

Keywords: signature, forward security, threshold, proactive.

1 Introduction

Proactive cryptography combines the concepts of "distributing the secret" and "refreshing the shares" to provide security against the mobile adversary, who attacks the parties of a distributed cryptosystem dynamically. For an adversary, we cannot assume that it cannot break into a particular party, who holds a share of the secret, during the party's lifetime. However, we can assume that the adversary can break into at most t parties during a short period of time, say an hour. Based on this observation, the proactive cryptography "refreshes" each party's share periodically. It divides the time into time periods, starting at 0. At the end of each time period, there is a "refresh phase" during which each party refreshes its share, but the secret they share remains intact. We assume that the mobile adversary can corrupt all parties during the lifetime of the cryptosystem; nevertheless, it can corrupt at most t parties during a time period. The proactive mechanism provides a high level of security for cryptosystems so that we would like to proactivize important cryptographic primitives.

In this paper we are interested in proactivizing the forward-secure signature scheme of Abadalla and Reyzin [3]. The Abadalla and Reyzin's forward-secure signature scheme (See Appendix) is an improvement of the Bellare-Miner

* Research supported in part by the National Science Council grant NSC-89-2213-E-009-180 and by the Ministry of Education grant 89-E-FA04-1-4, Taiwan, ROC.

K. Kim (Ed.): PKC 2001, LNCS 1992, pp. 264–276, 2001.

scheme [5] with a shorter public key. Abadalla, et. al. has proactivized the Bellare-Miner forward-secure signature scheme [2]. They proposed two threshold signature schemes in proactivizing Bellare-Miner forward-secure signature scheme. One scheme uses multiplicative secret sharing and the other uses polynomial secret sharing. In our scheme, we combine both secret sharing schemes for efficiency. We use multiplicative secret sharing in signing a message in threshold and polynomial secret sharing in sharing the signing secret. Our scheme is not only robust, but also efficient.

It is worth mentioning that we propose a new scheme for multiplying two secrets that are shared among parties [4,7,17]. Our multiplication scheme is efficient since it uses the public channel and the private channel once only.

2 Preliminaries

Communication model. We assume that the involved n parties are connected by a broadcast channel such that the messages over the channel cannot be blocked, delayed or altered. Nevertheless, one can inject false messages. Any two parties are connected by a private channel such that a third party cannot get messages sent over the private channel. We also assume that the communication channel is synchronous by rounds, that is, all parties send messages simultaneously in a round.

Time. There is a universal clock such that each party knows the absolute time. Therefore, we can divide time into time periods, starting at 0. Each time period has two phases: the execution phase and the refresh phase. The refresh phase follows the execution phase. The parties sign messages during the execution phase. During the refresh phase, all parties together run the share refresh algorithm to refresh their shares.

Adversary. We consider the static adversary who chooses corrupted parties at the beginning of each time period. The adversary runs three phases: the chosen message attack phase (CMA), the break-in phase (BREAKIN), and the forgery phase (FORGE). The BREAKIN phase for the threshold signature scheme is equivalent to the OVERTHRESHOLD phase of [2].

In the CMA phase, the adversary can corrupt at most t parties for any period of time. The adversary gets all information in the corrupted parties, including their shares, random bits, etc. The adversary can query the signing oracle $\mathcal{S}_x$, where x is the secret signing key. Since we assume the random oracle model [6], the adversary is allowed to query the random oracle $\mathcal{H}$ corresponding to the collision-resistant hash function used in the scheme. At the end of the CMA phase, the adversary can stay in the current phase or enter the next BREAKIN phase. In the BREAKIN phase, the adversary can corrupt *more than* t parties. Let c be the period that the adversary enters the BREAKIN phase and corrupts more than t users. In this phase, the adversary can compute the master secret (the signing key) of period c from the shares of corrupted parties. Then, the adversary enters the FORGE phase, during which the adversary outputs a forged signature of a new message which has not been queried to the signing oracle. We

say that the adversary succeeds in attacking the scheme if it outputs a forged signature for a prior time period c', $c' < c$, with non-negligible probability.

Forward security. A signature in the basic signature scheme is independent of time. Once the secret signing key is exposed, one can sign arbitrary messages. For forward-secure signature, the signing key evolves along time periods. At time period j, the signing key is SK_j. In the next time period $j+1$, the signing key is updated to SK_{j+1} and SK_j is deleted immediately. Although, the signing key evolves, the public key is the same for the whole lifetime. If one gets the signing key SK_c of time period c, he can fake the signatures of later time periods, but cannot fake the signatures of earlier time periods. Bellare and Miner [5] proposed the first forward-secure signature scheme based on difficulty of computing the square roots modulus a Blum integer. The scheme is actually converted from Fiat and Shamir's identification scheme [13]. To achieve security strength of level l, their scheme uses l public keys and l secret keys. Later, Abdalla and Reyzin [3] proposed an improvement based on the 2^l-th root problem [16,21,23,22]. With the same level of security strength, their scheme uses one public key and one secret key only.

Proactive security. Ostrovsky and Yung [24] proposed proactive security for distributed cryptographic schemes to deter mobile adversaries. For proactive security, the share in each party is refreshed at the end of each time period, but the signing secret key the parties share is unchanged at all time. A proactive cryptosystem remains secure as long as the adversary does not corrupt more than t parties in each time period. The shares of corrupted parties become useless when time enters the next time period. There is much literature about proactive cryptosystems [1,14,15,19,18,10,11,25,12].

3 Building Blocks

The following system setting is used throughout the rest of the paper.

- Let p=$4p'q'+1$ be a prime, where p' and q' are large primes and $p' \equiv q' \equiv 3 \pmod 4$.
- Let $N = p'q'$ and g a generator of the order-N subgroup of Z_p^*. All operations hereafter will be over the order-N subgroup, unless stated otherwise.
- The involved parties are dealers D_i, $1 \le i \le n$.

(t, n)-VSS PROCEDURE. If dealer D_i wants to share a random secret with other dealers, it runs the following steps.

1. Select a random polynomial $f_i(x)$ of degree t over Z_N^*. The constant coefficient of $f_i(x)$ is the random secret.
2. Send share $f_i(j)$ to dealer D_j, $j \neq i$ via the private channel.
3. Publish the verification values $\langle g^{a_{i,0}}, \ldots, g^{a_{i,t}} \rangle$.
4. Dealer D_j verifies validity of its received share $f_i(j)$ by $\prod_{k=0}^{k=t} g^{a_{i,k} j^k} = g^{f_i(j)}$.

If the verification fails, D_j requests D_i to publish $f_i(j)$. If D_i does not cooperate or posts an inconsistent $f_i(j)$, D_i is disqualified.

RECOVERY PROCEDURE. We use Lagrange's interpolation method to recover the secret with at least $t+1$ shares.

PROOF-SS PROCEDURE. Given (g, t, N, F, T), prover P wants to convince verifier V two things: (1) $a = \log_g F \bmod p = T^{1/t} \bmod N$ and (2) it knows this a. This is a combination of proofs of membership and knowledge.

1. The prover P selects random $w \in Z_N^*$ and sends $H = F^w$ and $B = w^t \bmod N$ to V.
2. The verifier V selects a random challenge $c \in \{0, 1\}$ and sends it to P.
3. The prover P sends $r = a^c w \bmod N$ to V.
4. The verifier V checks (1) $H = F^r$ and $B = r^t \bmod N$ if $c = 0$; and (2) $H = g^r$ and $B = r^t/T \bmod N$ if $c = 1$.

We use PROOF-SS(g, t, g^a, a^t) to denote the above interactive proof system.

Theorem 1. PROOF-SS *is complete, sound and zero-knowledge.*

Proof. (Sketch) The completeness property can be verified easily. For soundness of proof of knowledge, if any prover P^* can convince V with a non-negligible probability ϵ, P^* and V together can compute a with an overwhelming probability. By a probabilistic argument, there is a set W of w's of probability $\epsilon/2$ such that for every $w \in W$, P^* can answer two different challenges c_1 and c_2 with probability $\epsilon/2$. Therefore, we can get two responses $r_1 = a^{c_1} w \bmod N$ and $r_2 = a^{c_2} w \bmod N$ for the same commitments H and B. We can compute $a = r_2/r_1 \bmod N$ assuming, without loss of generality, $c_1 = 0$ and $c_2 = 1$. For soundness of proof of membership, we can easily show that if F and T are not of right form, the probability that P^* can cheat V is 0.5 (and is negligible after a polynomial number of rounds.)

For zero-knowledge, we construct a simulator S to simulate the view of any verifier V^*. S first selects $c' \in \{0, 1\}$ and $r \in Z_N^*$ randomly and computes $H = F^r$ and $B = r^t \bmod N$ if $c' = 0$ and $H = g^r$ and $B = r^t/T \bmod N$ if $c' = 1$. S then simulates $V^*(H, B)$ to get c. If $c = c'$, S outputs (H, B, c, r); otherwise S outputs $\perp$. The output of S conditioned on that the output is not $\perp$ and the view of V^* are statistically indistinguishable. □

We convert PROOF-SS into a non-interactive version by using a collision-resistant hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^l$ to replace V [9], where l be the security parameter. The message $(c, r_1, \cdots, r_l)$ sent by P for non-interactive PROOF-SS, denoted by NIPROOF-SS, satisfies

$$c = \mathcal{H}(t||g||N||F||T||H_1||B_1|| \cdots ||H_l||B_l),$$

where $||$ is the concatenation operator of strings. Let c_i denote the i-th bit of c. If c_i is 1, $H_i = g^{r_i}$ and $B_i = r_i^t/T \bmod N$; otherwise $H_i = F^{r_i}$ and $B_i = r_i^t \bmod N$. P can compute $(c, r_1, \cdots, r_l)$ by choosing $w_i \in Z_N^*$ for $i = 1, \cdots, l$, computing $c = \mathcal{H}(t||g||N||F||T||F^{w_1}||{w_1}^t|| \cdots ||F^{w_l}||{w_l}^t)$, and setting $r_i = a^{c_i} w_i \bmod N$. NIPROOF-SS releases no useful information under the random oracle model assuming hardness of discrete logarithm and factoring.

PROOF-DH PROCETURE. Given (g, H, F) and the prover P wants to convince V that $H = g^s$ and $F = g^{s^2}$ are of right form and it knows the secret s. The interactive proof system is as follows [8].

1. P randomly selects $w \in Z_N^*$ and sends $A = g^w$ and $B = H^w$ to V.
2. V sends a random challenge $c \in \{0, \cdots, 2^k - 1\}$ to P.
3. P sends the response $r = w + cs \bmod N$ to V.
4. V checks $g^r = A \cdot H^c$ and $H^r = B \cdot F^c$.

The above PROOF-DH procedure is complete, sound, and zero knowledge. We use NIPROOF-DH to denote its non-interactive version.

SQ PROCEDURE. Let $h(x)$ be a degree-t polynomial over Z_N^* with $h(0) = s$ and shared by the dealers D_i, $1 \le i \le n$. SQ's goal is to make the dealers share a degree-t polynomial $h'(x)$ over Z_N^* with $h'(0) = s^2 \bmod N$. SQ procedure is as follows.

1. Dealer D_i selects two degree-t polynomials $f_i(x)$ and $e_i(x)$ over Z_N^* at random, where $e_i(0)$=0. It sends shares $f_i(j)$ and $e_i(j)$ to D_j via the private channel, $1 \le j \le n$. Using (t, n)-VSS PROCEDURE, D_j checks if the received shares are correct. If so, all dealers share two degree-t polynomial $F(x) = \sum_{i=1}^n f_i(x) \bmod N$ and $E(x) = \sum_{i=1}^n e_i(x) \bmod N$. Each dealer D_i holds shares $F(i)$ and $E(i)$.
2. Each dealer D_i publishes $u_i = h(i)^2 + F(i) \bmod N$ and NIPROOF-DH$(g, g^{h(i)}, g^{h(i)^2})$ and checks validity of the published values of other dealers by checking $g^{u_j} = g^{h(j)^2} \cdot \prod_{k=0}^t g^{A_k j^k}$, where $g^{A_0}, g^{A_1}, \ldots, g^{A_t}$, computed from the verification values of $f_i(x)$'s, are the verification values of $F(x)$.
3. Each dealer D_i computes the degree-$2t$ polynomial $T(x)$=$h(x)^2 + F(x) = \sum_{k=0}^{2t} t_k x^k$ over Z_N from u_j, $1 \le j \le n$. Let $T''(x) = \sum_{k=0}^t t_k x^k$, which is $h''(x) + F(x) \bmod N$ for some degree-t polynomial $h''(x)$. Note that $h''(0) = h(0)^2 \bmod N$.
4. Each dealer D_i computes its share $h''(i) = T''(i) - F(i) \bmod N$ and randomizes it to become $h'(i) = h''(i) + E(i)$. The hidden polynomial becomes $h'(x) = h''(x) + E(x) \bmod N$ whose constant coefficient is still $s^2 \bmod N$.

We use SQ$(C, h(x), h'(x))$ to denote the above procedure, where C is the dealer set, $h(x)$ is the shared polynomial initially and $h'(x)$ is the shared polynomial at the end.

Theorem 2. SQ PROCEDURE *is correct, robust and secure if there are at most* $n/3$ *corrupted dealers.*

Proof. (Sketch) We can check correctness easily. Since there are at most t corrupted dealers, $t < n/3$, honest dealers can smoothly finish the procedure. This is guaranteed by the (t, n)-VSS procedure.

We present a simulator to show that a malicious adversary, who corrupts at most t dealers, gets no information. Let $\mathcal{B}$ be the corrupted set of dealers.

Input: $\langle g^s, g^{a_1}, \cdots, g^{a_t} \rangle$, $h(i)$ for every dealer $D_i \in \mathcal{B}$, where $h(x) = s + \sum_{k=1}^t a_k x^k$;

1. Randomly select degree-t polynomials $\hat{f}_i(x)$ and $\hat{e}_i(x)$ with $\hat{e}_i(0) = 0$, $1 \leq i \leq n$. Let $\hat{F}(x) = \sum_{i=1}^{n} \hat{f}_i(x)$ and $\hat{E}(x) = \sum_{i=1}^{n} \hat{e}_i(x)$.
2. Run (t, n)-VSS PROCEDURE.
3. For each $D_i \notin \mathcal{B}$, randomly select $\hat{u}_i$ over Z_N^*, compute $g^{h(i)^2} = g^{\hat{u}_i}/g^{\hat{F}(i)}$, and simulate NIPROOF-DH$(g, g^{h(i)}, g^{h(i)^2})$, where $g^{h(i)} = g^s \cdot \prod_{j=1}^{t} g^{a_j i^j}$.
4. For each $D_i \in \mathcal{B}$, publish $u_i = h(i)^2 + \hat{F}(i) \bmod N$ and simulate NIPROOF-DH$(g, g^{h(i)}, g^{h(i)^2})$.

The above simulation produces a distribution computationally indistinguishable from that of the real run. □

Assume that the dealers share two degree-t polynomial $h_1(x)$ and $h_2(x)$ initially. We can modify the SQ procedure so that the dealers share a degree-t polynomial $h'(x)$ whose constant coefficient is $h_1(0)h_2(0) \bmod N$ at the end. Let MULT$(C, h_1(x), h_2(x), h'(x))$ denote the procedure of sharing a degree-t polynomial $h'(x)$ whose constant coefficient is $h_1(0)h_2(0) \bmod N$.

4 Our Threshold Forward-Secure Signature Scheme

Our threshold forward-secure signature scheme, denoted by **TFSS**, is a key-evolving (t, s, n)-threshold signature scheme that consists of four procedures: TFSS.KEY, TFSS.UPDATE, TFSS.SIGN, and TFSS.VERIFY, where t is the maximum number of corrupted dealers, s is the minimum number of alive dealers so that signature computation is possible, and n is the total number of dealers. In our scheme, we set $s = t + 1$ and $n \geq 2t + 1$. There is a manager presiding the scheme.

TFSS.KEY: it generates the public key and each dealer D_i's initial secret-key share $S_{i,0}$ and public-key share $PK_{i,0}$ at time period 0.

1. Select N as that in the system setting.
2. The manager randomly selects $S_{i,0} \in Z_N^*$, $1 \leq i \leq n$ and sets $U_{i,0} = 1/S_{i,0}^{2^{l(T+1)}} \bmod N$, $S_0 = \prod_{i=1}^{n} S_{i,0} \bmod N$, and $U = 1/S_0^{2^{l(T+1)}} \bmod N$.
3. The system's initial secret key at time period 0 is SK_0=$(N, T, 0, S_0)$ and the public key $PK = (N, U, T)$.
4. Each dealer D_i's initial secret-key share is $SK_{i,0} = (N, T, 0, S_{i,0})$ and public-key share is $PK_{i,0} = (N, U_{i,0}, T)$.
5. Each dealer D_i shares its $S_{i,0}$ with other dealers by the (t, n)-VSS PROCEDURE.

TFSS.UPDATE: at the end of time period j, each dealer updates its secret-key and public-key shares from $S_{i,j}$ and $PK_{i,j}$ to $S_{i,j+1}$ and $PK_{i,j+1}$.

1. Each dealer D_i randomly selects $n-1$ numbers $s_{i,1}, s_{i,2}, \ldots, s_{i,n-1}$ over Z_N^* and computes $s_{i,n} = S_{i,j}/\prod_{k=1}^{n-1} s_{i,k} \bmod N$.
2. Each dealer D_i sends $s_{i,k}$ to D_k privately and publishes $\hat{s}_{i,k} = 1/s_{i,k}^{2^{l(T+1-j)}} \bmod N$, $1 \leq k \leq n$.

3. Each dealer D_k checks validity of the published values by $U_{i,j} = \prod_{r=1}^{n} \hat{s}_{i,r} \bmod N$, $1 \le i \le n, i \ne k$. Dealer D_k also checks validity of its received secret $s_{i,k}$ by $1/s_{i,k}^{2^{l(T+1-j)}} \bmod N = \hat{s}_{i,k}$. If any of the checks fails, all other dealers recover the secret $S_{i,j}$ by RECOVERY PROCEDURE.
4. Dealer D_i's new secret-key share is $S_{i,j+1} = (\prod_{k=1}^{n} s_{k,i})^{2^l} \bmod N$ and the corresponding public-key share is $U_{i,j+1} = \prod_{k=1}^{n} \hat{s}_{k,i} \bmod N$.
5. Dealer D_i shares $S_{i,j+1}$ with other dealers by (t,n)-VSS PROCEDURE. We use NIPROOF-SS$(g, t, g^{S_{i,j+1}}, S_{i,j+1}^t)$ to verify whether D_i's action is correct, where $t = -2^{l(T-j)}$ and $S_{i,j+1}^t = U_{i,j+1}$. If the proof is correct and (t,n)-VSS PROCEDURE succeeds, all dealers delete their old secret-key shares; otherwise, the secret of D_i is reconstructed.

TFSS.SIGN: at time period j, all dealers sign a messages M in a distributed way with the following steps.

1. Each dealer D_i selects $R_i \in Z_N^*$ randomly and publishes $Y_i = R_i^{2^{l(T+1-j)}} \bmod N$ and NIPROOF-SS$(g, 2^{l(T+1-j)}, g^{R_i}, Y_i)$. Then, it shares R_i to other dealers via (t,n)-VSS PROCEDURE with polynomial $f_i(x)$. If NIPROOF-SS or (t,n)-VSS PROCEDURE fails, set $R_i = 1$ and run RECOVERY PROCEDURE to recover the secret-key share $S_{i,j}$ of D_i.
2. Each dealer D_i computes $Y = \prod_{i=1}^{n} Y_i$ and $\sigma = \mathcal{H}(j, Y, M)$ and publishes its partial signature $Z_i = R_i S_{i,j}^{\sigma} \bmod N$.
3. Each dealer D_i verifies validity of another dealer D_k's partial signature by computing

$$Y_i' = Z_i^{2^{l(T+1-j)}} U_{i,j}^{\sigma} \bmod N$$

   and checking whether Y_i' and Y_i are equal. If the verification fails, all other alive dealers run RECOVERY PROCEDURE to recover the secret-key share $S_{k,j}$ and R_k of D_k and compute the partial signature Z_k.
4. Combine all partial signatures as a signature (j, Z, σ) for M at time j, where $Z = \prod_{i=1}^{n} Z_i \bmod N$. All dealers erase their R_i's.

TFSS.VERIFY: We can use the public key $PK = (N, U, T)$ of the system to verify validity of a signature (j, Z, σ) for M.

1. If $Z = 0$, return '0'.
2. Otherwise, compute $Y' = Z^{2^{l(T+1-j)}} U^{\sigma} \bmod N$ and output '1' if and only if $\sigma = H(j, Y', M)$.

5 Security Analysis

In this section, we show the correctness and security of our proposed scheme.

Theorem 3 (Correctness). *Assume that $SK_j = (N, T, j, S_j)$ and $PK = (N, U, T)$ are key pairs of the system at time period j. Each dealer D_i holds the secret-key share $SK_{i,j} = (N, T, j, S_{i,j})$ and public-key share $PK_{i,j} = (N, U_{i,j}, T)$. If (j, Z, σ) is generated by* TFSS.SIGN *for M,* TFSS.VERIFY$(PK, j, Z, \sigma)) = 1$.

Proof. We have $S_j = \prod_{i=1}^{n} S_{i,j} \bmod N$, $U = \prod_{i=1}^{n} U_{i,j} \bmod N = \prod_{i=1}^{n} S_{i,j}^{-2^{l(T+1-j)}} \bmod N$, $Y = \prod_{i=1}^{n} R_i^{2^{l(T+1-j)}} \bmod N = \prod_{i=1}^{n} Y_i \bmod N$ and $Z = \prod_{i=1}^{n} Z_i \bmod N = \prod_{i=1}^{n} R_i S_{i,j}^{\sigma} \bmod N$. Since

$$\begin{aligned} Y' &= Z^{2^{l(T+1-j)}} U^{\sigma} \bmod N = \prod_{i=1}^{n} (R_i S_{i,j}^{\sigma})^{2^{l(T+1-j)}} \prod_{i=1}^{n} U_{i,j}^{\sigma} \bmod N \\ &= \prod_{i=1}^{n} [R_i^{2^{l(T+1-j)}} S_{i,j}^{\sigma 2^{l(T+1-j)}} U_{i,j}^{\sigma}] \bmod N = \prod_{i=1}^{n} R_i^{2^{l(T+1-j)}} \bmod N \\ &= \prod_{i=1}^{n} Y_i \bmod N = Y, \end{aligned}$$

we have $\mathcal{H}(j, Y', M) = \mathcal{H}(j, Y, M) = \sigma$. □

Theorem 4. TFSS.UPDATE *procedure is secure against malicious adversaries.*

Proof. (Sketch) We construct a simulator S to simulate TFSS.UPDATE procedure assuming existence of malicious adversaries. Let $\mathcal{B} = \{D_{b_1}, \ldots, D_{b_t}\}$ be the set of corrupted servers at current time j. For simplicity, the secrets of corrupted dealers are treated as inputs. S simulates each dealer D_i's behavior as follows.

Input: $PK = (N, U, T)$, $S_{b_k,j}, 1 \le k \le t$, $\hat{f}_k(b_i), 1 \le i \le t, 1 \le k \le n$, $PK_{i,j} = (N, U_{i,j}, T), 1 \le i \le n$, and $\langle g^{S_{i,j}}, g^{a_{i,1}} \cdots, g^{a_{i,t}} \rangle, 1 \le i \le n$;

1. Randomly select $\hat{s}_{i,1}, \ldots, \hat{s}_{i,(i-1)}, \hat{s}_{i,(i+1)}, \ldots, \hat{s}_{i,n-1}$ from Z_N^*, compute

$$1/(\hat{s}_{i,i}^{2^{l(T+1-j)}}) = U_{i,j} / \prod_{k=1, k\neq i}^{n} 1/(\hat{s}_{i,k}^{2^{l(T+1-j)}}) \bmod N,$$

   and publish $\hat{S}_{i,k} = 1/(\hat{s}_{i,k}^{2^{l(T+1-j)}}) \bmod N$ for $k = 1, \ldots, n$. Note that we do not know the value $\hat{s}_{i,i}$ for $D_i \notin \mathcal{B}$.
2. Randomly select polynomial $\hat{h}_i(x)$ over Z_N^*. Let $\hat{h}_i(0) = S'_{i,j+1}$, which is a random value in Z_N^* for $D_i \notin \mathcal{B}$. Simulate NIPROOF-SS$(g, -2^{l(T-j)}, g^{S'_{i,j+1}}, U_{i,j+1})$, where $U_{i,j+1} = \prod_{k=1}^{n} \hat{S}_{k,i} \bmod N$. For $D_i \in \mathcal{B}$, compute its new secret-key share $S_{b_k,j+1}$ by $\prod_{i=1}^{n} \hat{s}_{i,b_k}^{2^l} \bmod N$ and simulate NIPROOF-SS$(g, -2^{l(T+1-(j+1))}, g^{S_{b_k,j+1}}, U_{b_k,j+1})$, where $U_{b_k,j+1} = \prod_{i=1}^{n} \hat{S}_{i,b_k} \bmod N$. Then, simulate (t, n)-VSS PROCEDURE.

If D_j forces D_i to disclose $\hat{s}_{i,j}$, since D_j has it, we can simulate $\hat{s}_{i,j}$. □

Theorem 5. *The* TFSS *scheme is a key-evolving (t, s, n)-threshold signature scheme for $s = t + 1$ and $n = 2t + 1$.*

Proof. (Sketch) Since there are at most t corrupted servers, their secret-key shares are not sufficient to recover the secret-key shares of honest dealers. The others follow the scheme. □

Theorem 6 (Forward secrecy). *Let* FS-DS *denote the single-user signature scheme in [3].* TFSS *is a threshold forward-secure signature scheme as long as* FS-DS *is a forward-secure signature scheme in the single-user sense.*

Proof. (Sketch) Let F be the adversary who attacks TFSS successfully by forging a signature (c', Z, α). We construct an algorithm that uses this F to forge a signature for the single-user FS-DS. As stated at Section 2, the attacking procedure contains three phases: CMA, BREAKIN, and FORGE. Our algorithm can query from the two oracles: the hashing oracle $\mathcal{H}$ and the singing oracle $\mathcal{S}$.

In the CMA phase, F guesses a particular time period c during which F breaks more than t dealers and gets the secret S_c. Let $U = 1/v^{2^{l(T+1-c)}}$ and $PK = (N, U, T)$, where $v = S_c$. We randomly select $U_{i,0}, \cdots, U_{n-1,0} \in_R Z_N^*$ and compute public-key share $U_{n,0} = U/\prod_{i=1}^{n-1} U_{i,0} \bmod N$. The public key is $PK_{i,0} = (N, U_{i,0}, T)$, $1 \le i \le n$. We simulate F by choosing a random tape for F, feeding all public keys to F, and running F in the CMA phase. F can corrupt at most t dealers except the time period c. Since F can corrupt at most t dealers except at time period c, we simply give all necessary secret-key shares and exchanged shares as F's input. F decides either to stay at the CMA phase or to switch to the BREAKIN phase, and then enter the FORGE phase.

We now we simulate the views of corrupted dealers during the key update phase. Let $\mathcal{B} = \{D_{b_1}, \cdots, D_{b_t}\}$ be the set of corrupted dealers at time period j. The simulation is the same as that of Theorem 4, which simulates the key update procedure. Note that the set of corrupted servers is decided in advance.

We can simulate the hash and signing oracles of F. For each query (j, Y, M) made by F, we query $\mathcal{H}$ on the same input and return the answer to F. We simulate the signing oracle of F by using $\mathcal{S}$. Let M be the message queried to $\mathcal{S}$. We give the direct answer (j, Z, σ) of S to F.

Now, we simulate F's view of the signing procedure. The input consists all secrets of the corrupted dealers and public information. For the input M and its signature $(j, Z, \sigma))$ seen by F, we construct the same probability distribution of F's real view as follows.

1. For $D_i \in \mathcal{B}$, we directly choose $R_i \in Z_N^*$ and publish $Y_i = R_i^{2^{l(T+1-j)}} \bmod N$ and NIPROOF-SS$(g, 2^{l(T+1-j)}, g^{R_i}, Y_i)$. Then, we simulate (t, n)-VSS PROCEDURE to share R_i with other dealers. Furthermore, we computes the partial signature $Z_i = R_i S_{i,j}^{\sigma} \bmod N$.
2. For $D_i \notin \mathcal{B}$, we computes its partial signature as follows. Let $Z' = Z/\prod_{i=1}^{t} Z_i \bmod N$. We randomly select $n-t-1$ numbers from Z_N^*, says $Z_{c_1}, \ldots, Z_{c_{n-t-1}}$. We compute $Z_{c_{n-t}} = Z'/\prod_{i=1}^{n-t-1} Z_{c_i} \bmod N$.
3. We compute $Y_{c_i} = Z_{c_i}^{2^{l(T+1-j)}} U_{c_i,j}^{\sigma} \bmod N$ for $1 \le i \le n-t$, and randomly select $(n-t)$ numbers from Z_N^*, says $R_{c_1}, \cdots, R_{c_{n-t}}$. We simulate NIPROOF-SS$(g, 2^{l(T+1-j)}, g^{R_{c_i}}, R_{c_i}^{2^{l(T+1-j)}})$ and run (t, n)-VSS PROCEDURE to share R_{c_i}, $1 \le i \le n-t$, with other dealers.
4. Finally, we compute $Y = \prod_{i=1}^{t} Y_{b_i} \prod_{j=1}^{n-t} Y_{c_j} \bmod N$ and sets $\mathcal{H}(j, Y, M) = \sigma$.

We can show that the above simulated view is statistically indistinguishable from the real view.

Obtaining a forgery. Let c be the time period that F switches to the BREAKIN phase. We provide the secret key S_c to F and run F to output a forged signature (c', Z, α) for M', where $c' < c$. (c', Z, α) is the forged signature for the single-user FS-DS, which is a contradiction. Therefore, our TFSS is secure. □

6 Discussion

Proactive security. We can easily add the proactive mechanism to TFSS.UPDATE. The only difference is to compute $\hat{s}_{i,k} = 1/s_{i,k}^{2^{l(T-j)}}$ in step 2, instead of $\hat{s}_{i,k} = 1/s_{i,k}^{2^{l(T-j+1)}}$, and new secret-key share $S'_{i,j} = \prod_{k=1}^{n} s_{k,i} \bmod N$ in the refresh phase. Furthermore, $s_{i,k}$ can be encrypted and sent to dealer D_k using D_k's public-key share $P_{k,j}$. This saves the private channel.

New construction. We can use polynomial secret sharing in our scheme, though it is less efficient. Our new construction is as follows. Initial setting is a bit different from that in Section 4. Let $f(x)$ be a degree-t polynomial with $f(0) = S_0$ and shared by all dealers by (t, n)-VSS PROCEDURE. To update the key S_j to S_{j+1}, all dealers compute the multiplication of two secrets for l times, where l is the security parameter. The robustness property is achieved by our MULT PROCEDURE. MULT PROCEDURE uses a proof to show that a dealer is honest. To compute a signature for a message, all dealers compute $l(T+1-j) + \log_2 \sigma$ times of distributed multiplication of secrets for $Y = R^{2^{l(T+1-j)}} \bmod N$ and $Z = RS_j^{\sigma} \bmod N$.

Efficiency. In our *new construction* based on polynomial secret sharing, dealers perform l multiplications of shares to update the key. That is, they exchange messages l times and compute l proofs for MULT PROCEDURE. To compute a signature, dealers exchange $l(T+1-j) + \log_2 \sigma$ messages and compute $l(T+1-j) + \log_2 \sigma$ proofs. As we can see, the computation and communication costs are quite expensive.

In *our main scheme* in Section 4, we combine the techniques of polynomial secret sharing and multiplicative secret sharing to reduce the cost. Each dealer exchanges messages twice in the key update stage, and once in the signing message stage. Each dealer needs to compute one proof in both key update and signing message stages. Therefore, our main scheme is quite efficient.

7 Conclusion

We have proposed a threshold forward-secure signature scheme, which is based on the 2^l-th root problem. Our scheme is robust and efficient in terms of the number of rounds so that the amount of exchanged messages among dealers is low. We show forward-secure security of our scheme based on that of the single-user scheme.

References

1. N. Alon, Z. Galil, M. Yung, "Dynamic-resharing verifiable secret sharing", *European Symposium on Algorithms 95 (ESA 95)*, Lecture Notes in Computer Science 979, pp.523-537, Springer-Verlag, 1995.
2. M. Abdalla, S. Miner, C. Namprempre, "Forward security in threshold signature schemes", manuscripts.
3. M. Abdalla, L. Reyzin, "A new forward-secure digital signature scheme", *Proceedings of Advances in Cryptology – Asiacrypt 2000*, Springer-Verlag, 2000.
4. M. Ben-Or, S. Goldwasser, A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computations", *Proceedings of the 20th ACM Symposium on Theory of Computing*, pp.1-10, 1988.
5. M. Bellare, S. Miner, "A forward-secure digital signature scheme", *Proceedings of Advances in Cryptology – Crypto 99*, Lecture Notes in Computer Science 1666, pp.431-448, Springer-Verlag, 1999.
6. M. Bellare, P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols", *Proceedings of the First ACM Conference on Computer and Communications Security*, pp.62-73, 1993.
7. D. Chaum, C. Crepeau, I. Damgard, "Multiparty unconditionally secure protocols", *Proceedings of the 20th ACM Symposium on Theory of Computing*, pp.11-19, 1988.
8. D. Chaum, T. Pedersen, "Wallet databases with observers", *Proceedings of Advances in Cryptology – Crypto 92*, pp.90-105, 1992.
9. U. Feige, A. Fiat, A. Shamir, "Zero-knowledge proof of identity", *Journal of Cryptology*, Vol. 1, pp.77-94, 1988.
10. Y. Frankel, P. Gemmell, P. MacKenzie, M. Yung, "Optimal-resilience proactive public-key cryptosystems", *Proceedings of 38th Annual Symposium on Foundations of Computer Science*, pp.384-393, IEEE, 1997.
11. Y. Frankel, P. Gemmell, P. MacKenzie, M. Yung, "Proactive RSA", *Proceedings of Advances in Cryptology – Crypto 97*, Lecture Notes in Computer Science 1294, pp.440-454, Springer-Verlag, 1997.
12. Y. Frankel, P. MacKenzie, M. Yung, "Adaptively-secure optimal-resilience proactive RSA", *Proceedings of Advances in Cryptology – Asiacrypt 99*, Lecture Notes in Computer Science 1716, pp.180-194, Springer-Verlag, 1999.
13. A. Fiat, A. Shamir, "How to prove yourself: practical solutions to identification and signature problems", *Proceedings of Advances in Cryptology – Crypto 86*, Lecture Notes in Computer Science 263, pp.186-194, Springer-Verlag, 1986.
14. R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, "Robust threshold DSS dignatures", *Proceedings of Advances in Cryptology – Eurocrypt 96*, Lecture Notes in Computer Science 1070, pp.354-371, Springer-Verlag, 1996.
15. R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, "Robust and efficient sharing of RSA functions", *Proceedings of Advances in Cryptology – Crypto 96*, Lecture Notes in Computer Science 1109, pp.157-172, Springer-Verlag, 1996.
16. L. Guillou, J. Quisquater, "A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory", *Proceedings of Advances in Cryptology – Eurocrypt 88*, Lecture Notes in Computer Science 330, pp.123-128, Springer-Verlag, 1988.
17. R. Gennaro, M. Rabin, T. Rabin, "Simplified VSS and fast-track multiparty computations with applications to threshold cryptography", *Proceedings of the 17th ACM Symposium on Principles of Distributed Computing (PODC)*, 1998.

18. A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, M. Yung, "Proactive public key and signature systems", *Proceedings of the 4th ACM Symposium on Computer and Communication Security*, 1997.
19. A. Herzberg, S. Jarecki, H. Krawczyk, M. Yung, "Proactive secret sharing or: how to cope with perpetual leakage", *Proceedings of Advances in Cryptology – Crypto 95*, Lecture Notes in Computer Science 963, pp.339-352, Springer-Verlag, 1995.
20. I. Ingemarsson, G. Simmons, "A protocol to set up shared secret schemes without the assistance of a mutually trusted party", *Proceedings of Advances in Cryptology – Eurocrypt 90*, Lecture Notes in Computer Science 473, pp.266-282, Springer-Verlag, 1990.
21. S. Micali, "A secure and efficient digital signature algorithm," *Technical Report MIT/LCS/TM-501*, Massachusetts Institute of Technology, Cambridge, MA, 1994.
22. H. Ong, C. Schnorr, "Fast signature generation with a Fiat-Shamir like scheme", *Proceedings of Advances in Cryptology – Eurocrypt 90*, Lecture Notes in Computer Science 473, pp.432-440, Springer-Verlag, 1990.
23. K. Ohta, T. Okamoto, "A modification of the Fiat-Shamir scheme", *Proceedings of Advances in Cryptology – Crypto 88*, Lecture Notes in Computer Science 403, pp.232-243, Springer-Verlag, 1988.
24. R. Ostrovsky, M. Yung, "How to withstand mobile virus attacks", *Proceedings of the 10th ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 51-61, 1991.
25. T. Rabin, "A simplified approach to threshold and proactive RSA", *Proceedings of Advances in Cryptology – Crypto 98*, Lecture Notes in Computer Science 1462, pp.89-104, Springer-Verlag, 1998.
26. A. Shamir, "How to share a secret", *Communications of the ACM*, 22(11), pp.612-613, 1979.

Appendix

Abdalla and Reyzin's forward-secure signature scheme. It has four procedures: key generation, key update, signing and verification. Let k and l be security parameters, T the largest time period, and $H : \{0,1\}^* \to \{0,1\}^l$ a collision-resistant hash function.

Key generation: generate the initial secret key SK_0 and public key PK.

1. Select two large primes p and q such that $p \equiv q \equiv 3 \pmod 4$, $2^{k-1} \leq (p-1)(q-1)$, and $pq < 2^k$. Let $N = pq$.
2. Randomly select S_0 from Z_N^* and compute $U = 1/S_0^{2^{l(T+1)}} \bmod N$.
3. Set the initial secret key $SK_0 = (N, T, 0, S_0)$ and the public key $PK = (N, U, T)$.

Key update: update the secret key SK_j to SK_{j+1}.

1. If $j = T$, set $SK_j = null$; otherwise, set $SK_{j+1} = (N, T, j+1, S_j^{2^l} \bmod N)$, where $SK_j = (N, T, j, S_j)$.

Signing: sign message M at time period j using key SK_j.

1. Randomly select $R \in Z_N^*$ and compute $Y = R^{2^{l(T+1-j)}} \bmod N$, $\sigma = H(j, Y, M)$, and $Z = RS_j^{\sigma} \bmod N$.
2. The signature is (j, Z, σ).

Verification: verify validity of signature (j, Z, σ) for M.

1. If $Z \equiv 0$, return 0; otherwise compute $Y' = Z^{2^{l(T+1-j)}} U^{\sigma} \bmod N$.
2. Output 1 if and only $\sigma = H(j, Y', M)$.