How to Construct Forward Secure Single-Server, Multi-Server and Threshold-Server Assisted Signature Schemes Using Bellare-Miner Scheme

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

Fanyu Kong[2], Rong Hao[1], Dexiang Zhang[3], Guowen Li[4]
[2]Institute of Network Security, Shandong University, Jinan, P. R. China
[3]Network Center, Qingdao University, Qingdao, P. R. China
[4]School of Computer Science and Technology, Shandong Jianzhu University, Jinan, P. R. China
Email: {sdukongfanyu, hr, dxzhang, gwl}@gmail.com

Abstract—**Server-assisted signature plays an important role in all kinds of applications in electronic commerce. It can be applied to the settings where a user employs public network servers to help her execute digital signature operations. In this paper, we discuss a problem of how to construct forward secure single-server, multi-server and threshold-server assisted signature schemes using Bellare-Miner Scheme and propose three signature schemes. In the single-server assisted signature scheme, the user (signer) can employ a server to help her produce a signature. In the multi-server assisted scheme, the number of employed servers is increased to *n*, therefore, it is infeasible for an attacker to forge a valid signature if she can't corrupt all servers and the signer. In the threshold-server assisted signature scheme, the generation of a valid signature needs the cooperation of the user and a quorum of servers. The scheme is more robust because the system can still get the correct signature as soon as no more than threshold servers are corrupted. These three schemes maintain the forward secure property, that is, even if the current secret key is exposed, an adaptive chosen-message adversary can't forge any signature pertaining to previous time period. Finally, we prove the proposed schemes are correct and forward secure.**

Index Terms—**digital signature, threshold signature, secret sharing, forward security, security analysis**

I. INTRODUCTION

Digital signatures as fundamental and useful building blocks of security protocols have wide applications in all kinds of electronic systems. Digital signature can provide a way for a user to sign documents so that the signatures can be verified by everyone. In a signature scheme, the user holds a secret (signing) key, while the public key is published so that everyone can know it. The user uses the secret key to create a signature for a message, but anyone can verify the validity of the signature using the corresponding public key. A digital signature provides three essential security services: authentication, integrity, and nonrepudiation. However, if the secret key to create the signature is stolen, then all the signatures pertaining to this secret key will not be trustworthy any more even if the signatures are created before the secret key is stolen. It is a very serious issue for some important applications such as Certificate Authority (CA) and electronic bank.

Forward secure signature is a solution to mitigate the damage of secret key leaks. The total time is divided into multiple time periods, and the different secret keys are used in different time periods while the public key remains fixed. Each new secret key is computed from the previous one via a one-way key update algorithm. Therefore, the compromise of one secret key in a certain period does not affect the security of the signatures signed in previous periods. Server-assisted signature is another effective signature to protect the security of the secret key. The secret key is shared between a user (signer) and a server, so it needs to compromise the user and the server simultaneously for an attacker to forge a valid signature.

Forward security for digital signature was firstly proposed by Anderson in an invited lecture of the 4th ACM conference on computer and communications security [1]. Bellare and Miner [2] formalized the forward secure signature and gave the first practical scheme. And then a sequence of forward secure signature schemes [3~6] with different merits were put forwards. Malkin *et al.* [7] proposed a forward secure signature scheme in which the operations of all sub-algorithms are independent of the total time periods *T* but liner to the current time period. How to construct a forward secure signature scheme with higher efficiency has been a hot topic in research for a long time [8]. Bilinear pairings advocated by Boneh and Franklin [9] could be used to construct forward secure signature schemes [12,13,14]. A fine-grained forward-secure signature scheme was proposed in [15], which allowed the signer to specify

which signatures of the current time period remain valid when revoking the public key. Forward secure signature schemes in untrusted update environments were researched in [16,17]. Forward secure threshold signature was also considered in [18~21].

A server-assisted signature was proposed by Xu *et al.* [22]. In their proposal, the signing key is shared among several servers and a user. The signing generation needs the cooperation of the user and a quorum number of servers. It is specially useful when a user employs several public network servers to help her execute threshold signature. Yang *et al.* [23] presented a proactive secret sharing for server assisted threshold signatures. Their scheme has higher security property, the proactive security. The paper [24] introduced an server assisted one-time signature alternative to server assisted signature scheme. The paper [25] revisited server assisted computation for digital signatures.

In this paper, we discuss a problem of how to construct forward secure single-server, multi-server and threshold-server assisted signature scheme using Bellare-Miner scheme. We construct three schemes with different properties. In the first scheme, the system is composed of one user and one server that share the secret key together. The user can employ the server to help her to produce a signature. The user and the server must cooperate to produce a valid signature. In the second scheme, the number of employed servers is increased to n, therefore, the signer must combine all n servers to produce a valid signature. It is infeasible for an attacker to forge a signature if she can't corrupt all servers and the user. In the last scheme, any generation of a valid signature needs the cooperation of the user and a quorum of servers. The scheme is more robust because the system can still get the correct signature as soon as no more than threshold servers are corrupted. The three schemes all maintain the forward secure property, that is, even if the current secret key is exposed, an adaptive chosen-message adversary can't forge any signature pertaining to previous time period. At last, the correctness and security have been analyzed.

The rest of the paper is organized as follows. In section 2, we introduce the preliminaries of our work including some definitions and robustness sub-protocols. The detailed description of schemes is given in Section 3. We analyze the correctness and forward security in section 4. Finally, we conclude this paper in the last section.

II. PRELIMINARIES

A. Definitions

Below we give the definition of forward secure server-assisted signature. Forward secure server-assisted signature consists of a key generation algorithm, a key update algorithm, a signing algorithm and a verifying algorithm. It satisfies that an adversary can't forge a signature of some new message for some previous period even if she gets the current private key.

Definition 1 (Forward Secure Server-Assisted Signature). A forward secure server-assisted signature is an quadruple of algorithms, *SAFSS*=(*SAFSS.GEN*, *SAFSS.UPD*, *SAFSS.SIG*, *SAFSS.VER*), where:

SAFSS.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, the initial share $SK_U^{(0)}$ for user U, and the share $SK_i^{(0)}$ for each server i(i=1,2,…,n).

SAFSS.UPD: the secret key update algorithm, inputs the current time period j, and generates a share $SK_i^{(j+1)}$ for each server i and $SK_U^{(j+1)}$ for user U in the algorithm for the next time period.

SAFSS.SIG: the signing algorithm, inputs the current time period j and a message M, and the user and all servers jointly generate a signature <j, tag> of M for period j.

SAFSS.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.

If <j, tag> is a valid signature generated in algorithm *SAFSS.SIG*, then *SAFSS.VER*(M, <j, tag>)=1.

When the number of servers is 1, we call this scheme as forward secure single-server assisted signature. When the number of servers is over 1, we call this scheme as forward secure multiple-server assisted signature.

Definition 2 (Forward Secure Threshold-Server Assisted Signature). A forward secure threshold-server assisted signature is an quadruple of algorithms, *SAFSTS*=(*SAFSTS.GEN*, *SAFSTS.UPD*, *SAFSTS.SIG*, *SAFSTS.VER*), where t-1 is the maximum number of corrupted servers; n is the total number of servers.

SAFSTS.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, the initial share $SK_U^{(0)}$ for user U, and the share $SK_i^{(0)}$ for each server i(i=1,2,…,n). Threshold servers can construct the partial secret key of all servers $SK^{(0)}$.

SAFSTS.UPD: the secret key update algorithm, inputs the current time period j, and generates a share $SK_i^{(j+1)}$ for each server i and $SK_U^{(j+1)}$ for user U in the algorithm for the next time period.

SAFSTS.SIG: the signing algorithm, inputs the current time period j and a message M, and the user and the set of threshold servers B($|B|=t$) jointly generate a signature <j, tag> of M for period j.

SAFSTS.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.

If <*j*, tag> is a valid signature generated in algorithm SAFSTS.SIG, then SAFSTS.VER(M, <*j*, tag>)=1.

B. The Robustness Sub-Protocols

(1) *Uncond-Secure-Z_N-VSS*: In this protocol, the shares of a secret are distributed to a group of players by a (*t, n*) secret sharing scheme in Z_N. It is information-theoretical secure and can tolerate (*n*-1)/2 malicious players. This protocol is taken from [26,27].

The dealer selects two random polynomials in Z_N: $f(x) \leftarrow \sum_{i=0}^{t} a_i x^i \bmod N$, $g(x) \leftarrow \sum_{i=0}^{t} b_i x^i \bmod N$, where $a_0 = L^2 s$, $s \in_R \{0,...,[n/4]-1\}$, $b_0 = L^2 k$, $k \in_R \{0,...,N^2([N/4]-1)\}$, $a_i, b_i \in_R \{0, L, 2L, ..., L^3 N^2\}$, $i = 1,...,t$. Compute $s_i = f(i)(i \in \{1...n\})$, $k_i = g(i)(i \in \{1...n\})$ and send (s_i, k_i) to players $i(i \in \{1...n\})$ secretly. Compute and broadcast the commits $C_i \triangleq g^{a_i} h^{b_i}, (i \in \{0,...,n\})$ for a_i . Players $i(i \in \{1...n\})$ verify $C(s_i, k_i) \stackrel{?}{\equiv} \prod_{j=0}^{t} (C_j)^{i^j}$. If the equation follows, the shares are right. Otherwise, publish (s_i, k_i) and generate a complaint against the dealer.

(2) *Jointly-Uncond-Secure-Z_N-RSS*: In this protocol, a group of players jointly and verifiably generate a random secret in Z_N by protocol *Uncond-Secure-Z_N-VSS* and each player holds one secret share. Each player can verify whether her share is right or not.

Each player *i* distributes a random secret $a_0^{(i)} \leftarrow L^2 \cdot s^{(i)}$ using protocol *Uncond-Secure- Z_N-VSS* as a deader role. Secretly send the share $s_j^{(i)}$ of secret $a_0^{(i)}$ and corresponding $k_j^{(i)}$ to player *j*. Broadcast commits $C_l^{(i)}(l \in \{0...n\})$. When player *j* receives the message, she checks the validity of the share using the equation $C(s_j^{(i)}, k_j^{(i)}) \stackrel{?}{\equiv} \prod_{l=0}^{t} (C_l^{(i)})^{jl}$. If it doesn't hold, player *j* publishes $(s_j^{(i)}, k_j^{(i)})$ and generates the complaint against player *i*. Suppose that the honest players constitute the set *A*. Note that all honest players get the same set *A*. The player *i* computes her share $s_i \leftarrow \sum_{j \in A} s_i^{(j)} \bmod N$.

(3) *Mult- Z_N -VSS*: In this protocol, compute the shares of the multiplication of two shared secrets in Z_N. At the same time, the exponent of the new polynomial doesn't increase (still *t*). This protocol combines the approaches in [28] and [32].

Each player *i* computes the following message: $\alpha_i = f_\alpha(i)$, $\beta_i = f_\beta(i)$, $\rho_i = \tau(i)$, $\sigma_i = s(i)$. Where the new polynomial satisfies the condition in [31]. The public message contains $\mathrm{A}_i = C(\alpha_i, \tau_i) = g^{\alpha_i} h^{\tau_i}$, $\mathrm{B}_i = C(\beta_i, \sigma_i) = g^{\beta_i} h^{\sigma_i}$.

Each player *i* executes DL-VSS protocol[29] to share $\lambda_i \alpha_i \beta_i$ like *Uncond-Secure-Z_N-VSS* protocol, where λ_i is the element of line 1 and column *i* in the inverse of Vandermonde matrix. Let $c_{ij} = f_{\alpha\beta i}(j), \tau_{ij} = u_i(j)$, where $f_{\alpha\beta i}$ and u_i are random polynomials in Z_N with order *t* and $f_{\alpha\beta i}(0) = \lambda_i \alpha_i \beta_i$.

All players employ VSPS-Check [32] protocol to check the validity of the share for player *i*. Player *i* proves $C_{i,0}$ is the valid commit about the multiplication of $\lambda_i \alpha_i \beta_i$ using ZK proof.

Player *i* computes $\gamma_i = \sum_{j=1}^{2t+1} c_{ji} \bmod N$ which is the share of $\gamma = \alpha\beta$ generated by a random polynomial with order *t*. Compute $\tau_i = \sum_{j=1}^{2t+1} \tau_{ji}$ and $C_j = C(\gamma_j, \tau_j) = g^{\gamma_j} h^{\tau_j} = \prod_{l=1}^{2t+1} C_{lj}$, $(1 \le j \le n)$ and broadcast $C_i (1 \le i \le n)$.

III. PRELIMINARIES

A. Forward Secure Single-Server Assisted Signature Scheme

The proposed forward secure single-server assisted signature is composed by one user and one server. There are four algorithms: algorithm *SAFSS.GEN*, algorithm *SAFSS.UPD*, algorithm *SAFSS.SIG*, algorithm *SAFSS.VER*. These algorithms are described as follows:

1. Algorithm *SAFSS.GEN*(*k,l,T*).
Begin
Firstly, select two random and distinct *k*/2-bit primes p_1, p_2 satisfying: $p_1 \equiv p_2 \equiv 3 (\mathrm{mod}\, 4)$, $N = p_1 p_2$.
For *i*=1 to *l* do
The dealer selects random values $S_{i,0}^s, S_{i,0}^U \in_R Z_N^*$.
Set $U_i^s = (S_{i,0}^s)^{2^{T+1}} \bmod N$,
$U_i^U = (S_{i,0}^U)^{2^{T+1}} \bmod N$, and $U_i = U_i^U \cdot U_i^s \bmod N$.
Endfor
Set $SK_0^s \leftarrow (N, T, 0, S_{1,0}^s, ..., S_{l,0}^s)$ and $SK_0^U \leftarrow (N, T, 0, S_{1,0}^U, ..., S_{l,0}^U)$.
Send SK_0^s to the server and send SK_0^U to the user secretly.
Set and publish $PK \leftarrow (N, T, U_1, ..., U_l)$.
End

2. Algorithm *SAFSS.UPD*(SK_j^s , SK_j^U , *j*).
Begin

Parse SK_j^s as $(N,T,0,S_{1,j}^s,...,S_{l,j}^s)$ and SK_j^U as $(N,T,0,S_{1,j}^U,...,S_{l,j}^U)$.

For *i*=1 to *l* do

Server computes $S_{i,j+1}^s=(S_{i,j}^s)^2 \bmod N$ and deletes $S_{i,j}^s$.

User computes $S_{i,j+1}^U=(S_{i,j}^U)^2 \bmod N$ and deletes $S_{i,j}^U$.

Endfor

Set the new shares of the server and the user $SK_{j+1}^s \leftarrow (N,T,0,S_{1,j+1}^s,...,S_{l,j+1}^s)$ and $SK_{j+1}^U \leftarrow (N,T,0,S_{1,j+1}^U,...,S_{l,j+1}^U)$.

End

3. Algorithm *SAFSS.SIG*(SK_j^s , SK_j^U ,*M, j*).

Begin

Parse SK_j^s as $(N,T,0,S_{1,j}^s,...,S_{l,j}^s)$ and SK_j^U as $(N,T,0,S_{1,j}^U,...,S_{l,j}^U)$.

The user selects random value $R_U \in_R Z_N^*$, computes and broadcasts $Y_U=(R_U)^{2^{(T+1-j)}} \bmod N$.

Server generates a random value $R_s \in_R Z_N^*$, computes and broadcasts $Y_s=(R_s)^{2^{(T+1-j)}} \bmod N$.

User and server jointly compute $Y=Y_U \cdot Y_S \bmod N$ and $c_1,c_2,...,c_l=H(j,Y,M)$.

User computes and broadcasts $Z_U=R_U\prod_{i=1}^{l}(S_{i,j}^U)^{c_i} \bmod N$.

Server computes and broadcasts $Z_s=R_s\prod_{i=1}^{l}(S_{i,j}^s)^{c_i} \bmod N$.

User and server cooperate to compute $Z=Z_U \cdot Z_s \bmod N$.

Generate signature <*j*, (*Y, Z*)> for message *M*.

End

4. Algorithm *SAFSS.VER*(*M*,<*j*, (*Y, Z*)>).

Begin

Parse sign as <*j*, (*Y, Z*)>

If $Y=0(\bmod N)$ then

Return 0

Else

$\sigma=H(j,Y,M)$

If $Y=Z^{2^{(T+1-j)}}U^{\sigma}(\bmod N)$ then

Return 1

Else

Return 0

End

B. Forward Secure Multi-server Assisted Signature Scheme

Below we introduce the second scheme. Different from the first scheme, the number of employed servers is increased to *n*. Therefore, the scheme has higher security than the first scheme because any adversary unable to corrupt the users and all *n* servers can't forge any valid signature. Even though she corrupts the user and all servers in a special time point, she can't forge any signature pertaining to previous time period.

1. Algorithm *SAFSS.GEN*(*k,l,n,T*).

Begin

Firstly, select two random and distinct *k*/2-bit primes p_1, p_2 satisfying: $p_1 \equiv p_2 \equiv 3(\bmod 4)$, $N=p_1p_2$.

For *i*=1 to *l* do

The dealer selects random value $S_{i,0}^{s_m}, S_{i,0}^U \in_R Z_N^*$, *m*=1,2,…,*n*,

Set $U_i^U=(S_{i,0}^U)^{2^{T+1}} \bmod N$,

$U_i^{s_m}=(S_{i,0}^{s_m})^{2^{T+1}} \bmod N$, *m*=1,2,…,*n*,

and $U_i=U_i^U \cdot \prod_{m=1}^{n} U_i^{s_m} \bmod N$.

Endfor

Set $SK_0^U \leftarrow (N,T,0,S_{1,0}^U,...,S_{l,0}^U)$ and $SK_0^{s_m} \leftarrow (N,T,0,S_{1,0}^{s_m},...,S_{l,0}^{s_m})$, *m*=1,2,…,*n*.

Send SK_0^U to the user and send $SK_0^{s_m}$ to server *m* secretly.

Set and publish $PK \leftarrow (N,T,U_1,...,U_l)$.

End

2. Algorithm *SAFSS.UPD*(SK_j^U , $SK_j^{s_1}$,...,$SK_j^{s_n}$, *j*).

Begin

Parse SK_j^U as $(N,T,0,S_{1,j}^U,...,S_{l,j}^U)$ and $SK_j^{s_m}$ as $(N,T,0,S_{1,j}^{s_m},...,S_{l,j}^{s_m})$.

For *i*=1 to *l* do

The user computes $S_{i,j+1}^U=(S_{i,j}^U)^2 \bmod N$, and deletes $S_{i,j}^U$.

Each server *m* computes $S_{i,j+1}^{s_m}=(S_{i,j}^{s_m})^2 \bmod N$, and deletes $S_{i,j}^{s_m}$.

Endfor

Set the new shares of the user and the servers $SK_{j+1}^U \leftarrow (N,T,0,S_{1,j+1}^U,...,S_{l,j+1}^U)$ and $SK_{j+1}^{s_m} \leftarrow (N,T,0,S_{1,j+1}^{s_m},...,S_{l,j+1}^{s_m})$, *m*=1,2,…,*n*.

End

3. Algorithm *SAFSS.SIG*(SK_j^U , $SK_j^{s_1}$,...,$SK_j^{s_n}$,*M, j*).

Begin

Parse SK_j^U as $(N,T,0,S_{1,j}^U,...,S_{l,j}^U)$ and $SK_j^{s_m}$ as $(N,T,0,S_{1,j}^{s_m},...,S_{l,j}^{s_m})$, *m*=1,2,…,*n*.

Each server *m* selects random value $R_{s_m} \in_R Z_N^*$, computes and broadcasts $Y_{s_m}=(R_{s_m})^{2^{(T+1-j)}} \bmod N$.

The user generates a random value $R_U \in_R Z_N^*$, computes and broadcasts $Y_U = (R_U)^{2^{(T+1-j)}} \bmod N$.

All servers and the user jointly compute $Y = Y_U \cdot \prod_{m=1}^{n} Y_{s_m} \bmod N$ and $c_1, c_2, ..., c_l = H(j, Y, M)$.

The server m computes and broadcasts $Z_{s_m} = R_{s_m} \prod_{i=1}^{l} (S_{i,j}^{s_m})^{c_i} \bmod N$.

The user computes and broadcasts $Z_U = R_U \prod_{i=1}^{l} (S_{i,j}^{U})^{c_i} \bmod N$.

All servers and the user cooperate to compute $Z = Z_U \cdot \prod_{m=1}^{n} Z_{s_m} \bmod N$.

Generate signature <j, (Y, Z)> for message M.

End

4. Algorithm *SAFSS.VER*(M,<j, (Y, Z)>).

Begin

Parse sign as <j, (Y, Z)>

If $Y = 0 (\bmod N)$ then

Return 0

Else

$c_1, ..., c_l = H(j, Y, M)$

If $Z^{2^{(T+1-j)}} = Y \cdot \prod_{i=1}^{l} U_i^{c_i} (\bmod N)$ then

Return 1

Else

Return 0.

End

C. Forward Secure Threshold-Server Assisted Signature Scheme

The third scheme is forward secure threshold-Server assisted signature scheme. This scheme is more robust because the dishonest behaviors of fewer than threshold servers don't affect the generation of valid signature. It is specially useful in the circumstance where the servers are easy to be attacked.

1. Algorithm *SAFSTS.GEN*(k,l,n,T).

Begin

Firstly, select two random and distinct k/2-bit primes p_1, p_2 satisfying: $p_1 \equiv p_2 \equiv 3 (\bmod 4)$, $N = p_1 p_2$.

For i=1 to l do

The dealer selects random value $S_{i,0}^{s_m}, S_{i,0}^{U} \in_R Z_N^*$, m=1,2,…,n,

Set $U_i^U = (S_{i,0}^U)^{2^{T+1}} \bmod N$,

Select $S_{i,0}^s \in_R Z_N^*$,

Compute $U_i^s = (S_{i,0}^s)^{2^{T+1}} \bmod N$

Generate the shares $S_{i,0}^{s_1}, S_{i,0}^{s_2}, ..., S_{i,0}^{s_n}$ of the partial secret $S_{i,0}^s$ using the protocol *Uncond-Secure-Z_N-VSS*.

and $U_i = U_i^U \cdot U_i^s \bmod N$.

Endfor

Set $SK_0^U \leftarrow (N, T, 0, S_{1,0}^U, ..., S_{l,0}^U)$ and $SK_0^{s_m} \leftarrow (N, T, 0, S_{1,0}^{s_m}, ..., S_{l,0}^{s_m})$, m=1,2,…,n.

Send SK_0^U to the user and send $SK_0^{s_m}$ to server m secretly.

Set and publish $PK \leftarrow (N, T, U_1, ..., U_l)$.

End

2. Algorithm *SAFSS.UPD*(SK_j^U, $SK_j^{s_1}$,...,$SK_j^{s_n}$, j).

Begin

Parse SK_j^U as $(N, T, 0, S_{1,j}^U, ..., S_{l,j}^U)$ and $SK_j^{s_m}$ as $(N, T, 0, S_{1,j}^{s_m}, ..., S_{l,j}^{s_m})$.

For i=1 to l do

The user computes $S_{i,j+1}^U = (S_{i,j}^U)^2 \bmod N$, and deletes $S_{i,j}^U$.

All servers jointly compute the new partial secret $S_{i,j+1}^s = (S_{i,j}^s)^2 \bmod N$ using protocol *Mult-Z_N-VSS*.

In doing so, each server m gets the new share $S_{i,j+1}^{s_m}$ and deletes the old share $S_{i,j}^{s_m}$.

Endfor

Set the new shares of the user and the servers $SK_{j+1}^U \leftarrow (N, T, 0, S_{1,j+1}^U, ..., S_{l,j+1}^U)$ and $SK_{j+1}^{s_m} \leftarrow (N, T, 0, S_{1,j+1}^{s_m}, ..., S_{l,j+1}^{s_m})$, m=1,2,…,n.

End

3. Algorithm *SAFSTS.SIG*(SK_j^U, $SK_j^{s_1}$,...,$SK_j^{s_n}$,M, j).

Begin

Parse SK_j^U as $(N, T, 0, S_{1,j}^U, ..., S_{l,j}^U)$ and $SK_j^{s_m}$ as $(N, T, 0, S_{1,j}^{s_m}, ..., S_{l,j}^{s_m})$, m=1,2,…,n.

All servers jointly generate a random value $R_s \in Z_N$ using protocol *Jointly-Uncond-Secure-Z_N-RSS* and each server m has its share R_{s_m}. Compute and publish $Y_s = (R_s)^{2^{(T+1-j)}} \bmod N$ using protocol $Mult - Z_N - VSS$.

The user generates a random value $R_U \in_R Z_N^*$, computes and broadcasts $Y_U = (R_U)^{2^{(T+1-j)}} \bmod N$.

All servers and the user jointly compute $Y = Y_U \cdot Y_s \bmod N$ and $c_1, c_2, ..., c_l = H(j, Y, M)$.

The server m computes and broadcasts $Z_s = R_s \prod_{i=1}^{l} (S_{i,j}^s)^{c_i} \bmod N$ using protocol *Mult-Z_N-VSS*.

The user computes and broadcasts $Z_U = R_U \prod_{i=1}^{l} (S_{i,j}^U)^{c_i} \bmod N$.

All servers and the user cooperate to compute $Z = Z_U \cdot Z_s \bmod N$.

Generate signature <j, (Y, Z)> for message M.

End

4. Algorithm *SAFSS.VER*(M,<j, (Y, Z)>).
Begin
Parse sign as <j, (Y, Z)>
If $Y = 0(\mathrm{mod}\, N)$ then
Return 0
Else
$c_1,...,c_l = H(j,Y,M)$
If $Z^{2^{(T+1-j)}} = Y \cdot \prod_{i=1}^{l} U_i^{c_i} (\mathrm{mod}\, N)$ then
Return 1
Else
Return 0.
End

IV. CORRECTNESS AND SECURITY ANALYSIS

We only give the correctness and security analysis of the forward secure multi-server assisted signature and forward secure threshold-server assisted signature. The analysis of forward secure single-server assisted signature scheme is very easy because it can be viewed as a special instance of the forward secure multi-server assisted signature scheme when n=1.

Theorem 1. (Correctness of *SAFSS*) Let the public key be $PK \leftarrow (N,T,U_1,...,U_l)$, $SK_0^U \leftarrow (N,T,0,S_{1,0}^U,...,S_{l,0}^U)$ and $SK_0^{s_m} \leftarrow (N,T,0,S_{1,0}^{s_m},...,S_{l,0}^{s_m})$, m=1,2,…,n. Algorithm *SAFSS.UPD* updates shares for each server and the user. Let <j, (Y, Z)> be the signature generated by algorithm *SAFSS.SIG* on input a message M for period j. Then *SAFSS.VER*(M, <j, (Y, Z)>)=1.

Proof. In order to verify whether *SAFSS.VER*(M, <j, (Y, Z)>)=1 holds or not, we need verify whether the equation $Z^{2^{(T+1-j)}} = Y \cdot \prod_{i=1}^{l} U^{c_i} (\mathrm{mod}\, N)$ follows. If the public key is $PK = (N,T,U)$, $SK_0^{s_m} \leftarrow (N,T,0,S_{1,0}^{s_m},...,S_{l,0}^{s_m})$, m=1,2,…,n. and $SK_0^U \leftarrow (N,T,0,S_{1,0}^U,...,S_{l,0}^U)$. Algorithm *SAFSS.UPD* updates shares for each server and the user, then $\sigma = H(j,Y,M)$, $Y = Y_U \cdot \prod_{m=1}^{n} Y_{s_m} \bmod N$,
$Y_U = (R_U)^{2^{(T+1-j)}} \bmod N$, $Y_{s_m} = (R_{s_m})^{2^{(T+1-j)}} \bmod N$,
$Z = Z_U \cdot \prod_{m=1}^{n} Z_{s_m} \bmod N$, $Z_U = R_U \prod_{i=1}^{l} (S_{i,j}^U)^{c_i} \bmod N$,
$Z_{s_m} = R_{s_m} \prod_{i=1}^{l} (S_{i,j}^{s_m})^{c_i} \bmod N$,
$U_i = U_i^U \cdot \prod_{m=1}^{n} U_i^{s_m} \bmod N$,
$S_{i,j+1}^U = (S_{i,j}^U)^2 \bmod N$,
$S_{i,j+1}^{s_m} = (S_{i,j}^{s_m})^2 \bmod N$.

Thus,

$$
\begin{aligned}
& Z^{2^{(T+1-j)}} \\
& \equiv (Z_U \cdot \prod_{m=1}^{n} Z_{s_m})^{2^{(T+1-j)}} \\
& \equiv (R_U \prod_{i=1}^{l} (S_{i,j}^U)^{c_i})^{2^{(T+1-j)}} \cdot (\prod_{m=1}^{n} (R_{s_m} \prod_{i=1}^{l} (S_{i,j}^{s_m})^{c_i}))^{2^{(T+1-j)}} \\
& \equiv R_U^{2^{(T+1-j)}} \cdot \prod_{m=1}^{n} (R_{s_m})^{2^{(T+1-j)}} \cdot \prod_{i=1}^{l} (S_{i,j}^U)^{2^{(T+1-j)} \cdot c_i} \\
& \quad \cdot \prod_{m=1}^{n} \prod_{i=1}^{l} (S_{i,j}^{s_m})^{2^{(T+1-j)} c_i} \\
& \equiv Y_U \cdot \prod_{m=1}^{n} Y_{s_m} \cdot \prod_{i=1}^{l} (S_{i,0}^U)^{2^{(T+1)} \cdot c_i} \cdot \prod_{m=1}^{n} \prod_{i=1}^{l} (S_{i,0}^{s_m})^{2^{(T+1)} c_i} \\
& \equiv Y_U \cdot \prod_{m=1}^{n} Y_{s_m} \cdot \prod_{i=1}^{l} (U_i^U)^{c_i} \cdot \prod_{i=1}^{l} \prod_{m=1}^{n} (U_i^{s_m})^{c_i} \\
& \equiv Y \cdot \prod_{i=1}^{l} (U_i^U \prod_{m=1}^{n} U_i^{s_m})^{c_i} \\
& \equiv Y \cdot \prod_{i=1}^{l} (U_i)^{c_i} \ (\mathrm{mod}\, N)
\end{aligned}
$$

Therefore, there is *SAFSS.VER*(M, <j, (Y, Z)>)=1

Theorem 2. (Correctness of *SAFSTS*) Let the public key be $PK \leftarrow (N,T,U_1,...,U_l)$, $SK_0^U \leftarrow (N,T,0,S_{1,0}^U,...,S_{l,0}^U)$ and $SK_0^{s_m} \leftarrow (N,T,0,S_{1,0}^{s_m},...,S_{l,0}^{s_m})$, m=1,2,…,n. Algorithm *SAFSTS.UPD* updates shares for each server and the user. Let <j, (Y, Z)> be the signature generated by algorithm *SAFSTS.SIG* on input a message M for period j. Then *SAFSTS.VER*(M, <j, (Y, Z)>)=1.

Proof. Similarly to the proof in theorem, in order to verify whether *SAFSTS.VER*(M, <j, (Y, Z)>)=1 holds or not, we need verify whether the equation $Z^{2^{(T+1-j)}} = Y \cdot \prod_{i=1}^{l} U^{c_i} (\mathrm{mod}\, N)$ follows. If the public key is $PK = (N,T,U)$, $SK_0^{s_m} \leftarrow (N,T,0,S_{1,0}^{s_m},...,S_{l,0}^{s_m})$, m=1,2,…,n. and $SK_0^U \leftarrow (N,T,0,S_{1,0}^U,...,S_{l,0}^U)$. Algorithm *SAFSTS.UPD* updates shares for each server and the user, then $\sigma = H(j,Y,M)$, $Y = Y_U \cdot Y_s \bmod N$,
$Y_U = (R_U)^{2^{(T+1-j)}} \bmod N$, $Y_s = (R_s)^{2^{(T+1-j)}} \bmod N$,
$Z = Z_U \cdot Z_s \bmod N$, $Z_U = R_U \prod_{i=1}^{l} (S_{i,j}^U)^{c_i} \bmod N$,
$Z_s = R_s \prod_{i=1}^{l} (S_{i,j}^s)^{c_i} \bmod N$,
$U_i = U_i^U \cdot U_i^s \bmod N$,
$S_{i,j+1}^U = (S_{i,j}^U)^2 \bmod N$,
$S_{i,j+1}^s = (S_{i,j}^s)^2 \bmod N$.

Thus,

$$
\begin{aligned}
& Z^{2^{(T+1-j)}} \\
& \equiv (Z_U \cdot Z_s)^{2^{(T+1-j)}} \\
& \equiv (R_U \prod_{i=1}^{l} (S_{i,j}^U)^{c_i})^{2^{(T+1-j)}} \cdot (R_s \prod_{i=1}^{l} (S_{i,j}^s)^{c_i})^{2^{(T+1-j)}} \\
& \equiv R_U^{2^{(T+1-j)}} \cdot R_s^{2^{(T+1-j)}} \cdot \prod_{i=1}^{l} (S_{i,j}^U)^{2^{(T+1-j)} \cdot c_i} \cdot \prod_{i=1}^{l} (S_{i,j}^s)^{2^{(T+1-j)} \cdot c_i} \\
& \equiv Y_U \cdot Y_S \cdot \prod_{i=1}^{l} (S_{i,0}^U)^{2^{(T+1)} \cdot c_i} \cdot \prod_{i=1}^{l} (S_{i,0}^s)^{2^{(T+1)} \cdot c_i} \\
& \equiv Y_U \cdot Y_S \cdot \prod_{i=1}^{l} (U_i^U)^{c_i} \cdot \prod_{i=1}^{l} (U_i^s)^{c_i} \\
& \equiv Y \cdot \prod_{i=1}^{l} (U_i)^{c_i} \ (\mathrm{mod}\, N)
\end{aligned}
$$

Therefore, there is $SAFSTS.VER(M, <j, (Y, Z)>)=1$

The security of our schemes base on the security of Bellare-Miner Scheme and Abdalla scheme [14], which in turn relies on the hardness of factoring.

Theorem 3 (Security of *SAFSS*) Assuming that the problem of factoring Blum-Williams integer is computationally intractable, it is computationally infeasible to break the forward security of our forward secure multi-server assisted signature.

Proof.

Let F be an adversary to attack the security of our scheme. F works in three phases: the chosen-message attack phase; the corrupt phase; and the forgery phase. Let *FTSIG* be the multiplicative secret sharing threshold signature scheme in [19]. We want to construct an algorithm I against the security of *FTSIG* using F as a subroutine. I works in three stages: the chosen-message attack phase; the break-in phase; and the forgery phase,

We can set $P = \{server_1, ..., server_n, user\}$ and let $t = n$, $P_i = server_i (i = 1, 2, ..., t)$, $P_{n+1} = user$. Then our scheme is a *FTSIG(t,t*+1, *t*+1, *t*+1) scheme for the group of players P. Because the views of adversary in *FSIG.key*, *FSIG.update*, and *FSIG.sign* can all be simulated, the views of adversary in *SAFSS.GEN*, *SAFSS.UPD*, and *SAFSS.SIG* can all be simulated in our scheme, too. Therefore, when I works in the chosen-message attack phase, we set F come to its chosen-message attack phase. When I works in break-in phase, we set F come to its corrupt phase and provide the corrupted private key to I. When I works in forge phase to forge a new signature, F can come to its forge phase to provide the same forgery. Because *FTSIG* scheme is forward secure assuming factoring is hard, therefore, our scheme is secure forward. It means the adversary cannot forge any signature of previous time period, when she gets the shares of the user and all servers in a time period.

Theorem 4 (Security of *SAFSTS*) Assuming that the problem of factoring Blum-Williams integer is computationally intractable, it is computationally infeasible to break the forward security of our forward secure threshold-server assisted signature.

There is an adversary attacks the security of *BMSIG* scheme if our scheme can be attacked by an algorithm. Because *BMSIG* scheme is forward secure assuming factoring is hard. Therefore, our scheme is forward secure in the random oracle under the same assumption. Here we skip the detailed proof of this theorem.

V CONCLUSIONS

We discuss a problem of how to construct forward secure single-server, multi-server and threshold-server assisted signature schemes using Bellare-Miner Scheme and propose three signature schemes in this paper. The first is single-server assisted scheme, the second is multi-server assisted scheme and the last is threshold-server assisted scheme. These schemes are all based on the Bellare-Miner forward secure signature Scheme. We also analyze the correctness and security. The extended abstract has appeared in International Symposium on Electronic Commerce 2008 [30].

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] R. Anderson, “Two remarks on public key cryptology,” Invited Lecture, 4th ACM Conference on Computer and Communications Security. Zurich, 1997.

[2] M. Bellare, and S. Miner, “A forward-secure digital signature scheme,” Advances in Cryptology-Crypto’99, 1999, pp. 431-448.

[3] M. Abdalla and L. Reyzin, “A new forward-secure digital signature scheme,” Advances in Cryptology-Asiacrypt’00, 2000, pp. 116-129.

[4] H. Krawczyk, “Simple forward-secure signatures for any signature scheme,” Proceedings of the 7th ACM Conference on Computer and Communications Security, 2000, pp. 108-115.

[5] G. Itkis, and L. Reyzin, “Forward-secure signatures with optimal signing and verifying,” Advances in Cryptology-Crypto’01, 2001, 499-514.

[6] A. Kozlov, and L. Reyzin, “Forward-secure signatures with fast key update,” Security in Communication Network, 2002, pp. 247-262.

[7] T. Malkin, D. Micciancio, and S. Miner, “Efficient generic forward-secure signatures with an unbounded number of time periods,” Advances in Cryptology-Eurocrypt’02, 2002, pp. 400-417.

[8] G. Itkis, “Forward Security: Adaptive Cryptography–Time Evolution,”http://www.cs.bu.edu/faculty/itkis/pap/forward-secure-survey. pdf, 2005.

[9] D. Boneh, and M. Franklin, “Identity-based encryption from the Weil pairing,” Advances in Cryptology-CRYPTO’001, 2001, pp. 213–229.

[10] C. Gentry, and A. Silverberg, “Hierarchical ID-based cryptography,” Advances in Cryptology-ASIACRYPT ‘02, 2002, pp. 548-566.

[11] R. Canetti, S. Halevi, and J. Katz, “A forward-secure public-key encryption scheme,” Advances in Cryptology-EUROCRYPT’03, 2003, pp. 255-271.

[12] F. Hu, C. Wu, and J. Irwin, “A new forward secure signature scheme using bilinear maps,” Cryptology ePrint Archive, Report 2003/188, 2003.

[13] B. Kang, J. Park, and S. Halm, “A new forward secure signature scheme,” Cryptology ePrint Archive, Report 2004/183, 2004.

[14] J. Yu, F. Y. Kong, X. G. Cheng, R. Hao, and G. W. Li, “Construction of Yet Another Forward Secure Signature Scheme Using Bilinear Maps,” In: the second international conference on provable security (ProvSec 2008), 2008, pp. 83-97.

[15] 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.
[16] X. Boyen, H. Shacham, E. Shen, and B. Waters, "Forward Secure Signatures with Untrusted Update," The 13th ACM conference on Computer and communications security, 2006, pp. 191-200.
[17] B. Libert, J. Jacques, and M. Yung, "Forward-Secure Signatures in Untrusted Update Environments: Efficient and Generic Constructions," The 14th ACM conference on Computer and communications security, 2007, pp. 266-275.
[18] M. Abdalla, S. Miner, and C. Namprempre, "Forward-secure threshold signature schemes," Topics in Cryptology–CT-RSA'01, 2001, pp. 441–456.
[19] Z. J. Tzeng, and W. G. Tzeng, "Robust forward signature schemes with proactive security," In Proceedings of the Public-Key Cryptography'01, 2001, pp. 264-276.
[20] 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.
[21] 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.
[22] S. Xu, and R. Sandhu. "Two efficient and provably secure scheme for server-assisted threshold signature," CT-RSA'03, 2003, pp. 355-373.
[23] J. Yang, K. Rhee, and K. Sakurai, "A proactive secret sharing for server assisted threshold signatures," HPCC '06, 2006, pp. 250-259.
[24] K. Bicakci, and N. Baykal, "SAOTS: A New Efficient Server Assisted Signature Scheme for Pervasive Computing," Proceedings of 1st International Conference on Security in Pervasive Computing SPC'03, 2003, pp. 187-200.
[25] K. Bicakci, and N. Baykal, "Server Assisted Signatures Revisited," The Cryptographer's Track at the RSA Conference, 2004, pp. 143-156.
[26] Y. Frankel, P.D. Mackenzie, and M. Yung, "Robust efficient distributed rsa-key generation," In Proceedings of the 30th Annual ACM Symposium on the Theory of Computing (STOC'98), 1998, pp. 663-672.
[27] C. Chu, L. Liu, and W. Tzeng, "A Threshold GQ Signature Scheme," http://eprint.iacr.org/2003/016.pdf
[28] Y. Frankel, P.D. Mackenzie, and M. Yung, "Adaptively-secure optimal-resilience proactive rsa," In Proceedings of Advances in Cryptology –ASIACRYPT'99, 1999, pp. 180-194.
[29] R. Gennaro, M. O. Rabin, and T. Rabin, "Simplified VSS and fast-track multiparty computations with applications to threshold cryptography," In 17th ACM Symposium Annual on Principles of Distributed Computing, Puerto Vallarta, Mexico, 1998, pp. 101-111.
[30] J. Yu, F. Y. Kong, R. Hao, "Construction of Server-assisted Forward Secure Signature Using Bellare-Miner Scheme". International Symposium on Electronic Commerce and Security (ISECS 2008), 2008, pp. 558-561.

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.

Rong Hao was born in China in 1976. He received the BS, MS degrees in computer science from Jinan University and Shandong University, Shandong, China, in 1998 and 2006, respectively.

She became a lecturer of computer science in the College of Information Engineering at Qingdao University, China, in 2006. She is currently a fellow in the College of Information Engineering at Qingdao University, China. Her research interests include cryptography and network security.