Sharing DSS by the Chinese Remainder Theorem

Kamer Kaya*,[a], Ali Aydın Selçuk[b]

[a] *Ohio State University, Columbus, 43210, OH, USA*
[b] *Bilkent University, Ankara, 06800, Turkey*

Abstract

In this paper, we propose a new threshold scheme for the Digital Signature Standard (DSS) using Asmuth-Bloom secret sharing based on the Chinese Remainder Theorem (CRT). To achieve the desired result, we first show how to realize certain other threshold primitives using Asmuth-Bloom secret sharing, such as joint random secret sharing, joint exponential random secret sharing, and joint exponential inverse random secret sharing. We prove the security of our scheme against a static adversary. To the best of our knowledge, this is the first provably secure threshold DSS scheme based on CRT.

Key words: Asmuth-Bloom secret sharing, threshold cryptography, function sharing, DSS

1. Introduction

Threshold cryptography deals with the problem of sharing a highly sensitive secret among a group of n users so that the secret can be reconstructed only when a sufficient number t of them come together. This problem is known as the secret sharing problem and several secret sharing schemes (SSS) have been proposed in the literature (e.g., [1, 3, 16]).

Threshold cryptography also deals with the function sharing problem. A function sharing scheme (FSS) requires distributing the function's computation according to the underlying SSS such that each part of the computation

*Corresponding Author: Tel: (614) 366-2476, Fax: (614) 688-6600
Email addresses: `kamer@bmi.osu.edu` (Kamer Kaya), `selcuk@cs.bilkent.edu.tr` (Ali Aydın Selçuk)

can be carried out by a different user and then the partial results can be combined to yield the function's value without disclosing individual secrets. The FSSes in the literature, e.g., [4, 5, 15, 17], proposed for various cryptosystems, traditionally used Shamir's SSS [16] until a recent work by Kaya and Selcuk [12] showed how to use the Asmuth-Bloom SSS [1] for function sharing. In that paper [12], they propose threshold RSA signature/encryption, ElGamal encryption, and Paillier encryption schemes by using the Asmuth-Bloom SSS.

The Digital Signature Standard (DSS) is the current US government standard for digital signatures. Sharing DSS is an interesting problem and a solution was given by Gennaro et al. [9], based on Shamir's SSS.

In this paper, we propose a new threshold scheme for DSS with the Asmuth-Bloom SSS. We follow the approach of Gennaro et al. [9] and show how similar primitives can be achieved based on Asmuth-Bloom secret sharing. Obtaining these primitives with Asmuth-Bloom secret sharing turns out to be a challenging task. We use the approximate-and-correct approach [12] to compute the correct function output values from the partial results. The proposed scheme is provably secure against up to $t-1$ corrupted users, and $2t+2$ users are required to generate a DSS signature. To the best of our knowledge, this is the first provably secure threshold DSS scheme based on the Chinese Remainder Theorem (CRT).

2. Digital Signature Standard

DSS [7] is based on the ElGamal signature scheme [6], where the ElGamal signature is slightly modified to obtain fixed-length outputs even though the size of the prime modulus p may increase. The DSS signature scheme can be summarized as follows:

- *Key Generation Phase:* Let p and q be large prime numbers, where $q|(p-1)$ and let $g \in \mathbb{Z}_p^*$ be an element of order q. The private key $\alpha \in_R \mathbb{Z}_q^*$ is chosen randomly and the public key $\beta = g^\alpha \bmod p$ is computed.

- *Signing Phase:* The signer first chooses a random ephemeral key $k \in_R \mathbb{Z}_q^*$ and then computes the signature (r, s), where

$$r = (g^{k^{-1}} \bmod p) \bmod q,$$
$$s = k(w + \alpha r) \bmod q$$

  for a hashed message $w \in \mathbb{Z}_q$.

- *Verification Phase:* The signature (r, s) is verified by checking $r \overset{?}{=} (g^{ws^{-1}} \beta^{rs^{-1}} \bmod p) \bmod q$, where s^{-1} is computed in $\mathbb{Z}_q^*$.

3. The Asmuth-Bloom Secret Sharing Scheme

The Asmuth-Bloom SSS [1] shares a secret d among n parties such that any t users can reconstruct the secret by the CRT. The scheme below is a slightly modified version by Kaya and Selcuk [12] in order to obtain better security properties.

- *Dealing Phase:* To share a secret d among a group of n users, the dealer does the following:
    1. A set of relatively prime integers $m_0 < m_1 < \ldots < m_n$ are chosen, where m_0 is a prime and
    $$\prod_{i=1}^{t} m_i > {m_0}^2 \prod_{i=1}^{t-1} m_{n-i+1}. \tag{1}$$
    2. Let M denote $\prod_{i=1}^{t} m_i$. The dealer computes $y = d + Am_0$, where A is a positive integer generated randomly such that $0 \le y < M$.
    3. The share of the ith user, $1 \le i \le n$, is $y_i = y \bmod m_i$.

- *Combining Phase:* Let S be a coalition of t users gathered to construct the secret. Let M_S denote $\prod_{i \in S} m_i$.
    1. Let $M_{S\setminus\{i\}}$ denote $\prod_{j \in S, j \neq i} m_j$ and $M'_{S,i}$ be the multiplicative inverse of $M_{S\setminus\{i\}}$ in $\mathbb{Z}_{m_i}$, i.e., $M_{S\setminus\{i\}} M'_{S,i} \equiv 1 \pmod{m_i}$. First, the ith user computes $u_i = y_i M'_{S,i} M_{S\setminus\{i\}} \bmod M_S$.
    2. The users compute $y = \left(\sum_{i \in S} u_i\right) \bmod M_S$ and then obtain the secret d by computing $d = y \bmod m_0$.

4. The Modified Asmuth-Bloom SSS

In the literature, the sequence, $m_0 < m_1 < \ldots < m_n$, satisfying (1), is called a (t, n) Asmuth-Bloom sequence. An interesting property of these sequences, which will be used in our scheme, is given in Lemma 1.

Lemma 1. *An $(\lceil n/2 \rceil, n)$ Asmuth-Bloom sequence is a (k, n) Asmuth-Bloom sequence for all k such that $1 \le k \le n$.*

To adapt the original scheme for the threshold DSS, first, we use such an $(\lceil n/2 \rceil, n)$ sequence. Note that this sequence can be used for any (k, n) Asmuth-Bloom SSS where $M = \prod_{i=1}^{k} m_i$. Second, we multiply the right side of (1) by n and obtain the inequality

$$\prod_{i=1}^{t} m_i > n{m_0}^2 \prod_{i=1}^{t-1} m_{n-i+1}. \tag{2}$$

Lastly, we change the definition of M to

$$M = \left\lfloor \frac{\prod_{i=1}^{t} m_i}{n} \right\rfloor. \tag{3}$$

Theorem 1 (Kaya and Selcuk [13]). *The modified Asmuth-Bloom scheme is perfect. I.e., an adversary with $t-1$ or fewer shares has $\Pr(d = d') = \Pr(d = d'')$ for any $d', d'' \in \mathbb{Z}_{m_0}$.*

4.1. Arithmetic Properties of the Modified Asmuth-Bloom SSS

Suppose several secrets are shared with common parameters t, n, and m_i, $1 \leq i \leq n$, chosen according to (2). Shareholders can use the following properties to obtain new shares for the sum and product of the shared secrets:

Proposition 1. *Let $d_1, d_2, \cdots, d_n$ be secrets shared by the Asmuth-Bloom SSS with common parameters t, n, and moduli m_i for $1 \leq i \leq n$. Let y_{ij} be the share of the ith user for secret d_j. Then, for $D = (\sum_{i=1}^{n} d_i) \bmod m_0$ and $Y_i = (\sum_{j=1}^{n} y_{ij}) \bmod m_i$, we have $D \overset{t}{\leftarrow} (Y_1, Y_2, \cdots, Y_n)$, i.e., by using t of $Y_1, Y_2, \cdots, Y_n$, the secret D can be constructed.*

Proposition 2. *Let d_1, d_2 be secrets shared by the Asmuth-Bloom SSS with common parameters t, n and moduli m_i for $1 \leq i \leq n$. Let y_{ij} be the share of the ith user for secret d_j. Then, for $D = d_1 d_2 \bmod m_0$ and $Y_i = y_{i_1} y_{i_2} \bmod m_i$, we have $D \overset{2t}{\leftarrow} (Y_1, Y_2, \cdots, Y_n)$.*

5. Sharing DSS

To obtain a threshold DSS scheme, first the dealer generates the private key α and shares it among the users by a (t, n) Asmuth-Bloom secret sharing scheme with $m_0 = q$. Then a signing coalition S can sign a message without requiring a trusted party. Note that anyone can obtain the secret key α and forge signatures if he knows k for a valid signature (r, s). Hence, $r = (g^{k^{-1}} \bmod p) \bmod q$ must be computed in a way that no one obtains k.

Our approach for this problem can be summarized as follows: First the users in the signing coalition S will share a random k value by using the joint random secret sharing primitive, JOINT-RSS, in which each user of S contributes to the sharing of k. Note that this procedure only generates the shares for k, not k itself. Next, the JOINT-EXP-INVERSE primitive is called to compute $r = (g^{k^{-1}} \bmod p) \bmod q$. Last, the signature will be obtained by using r and the shares of k and α.

Below, S denotes the signing coalition of size $2t + 2$. Without loss of generality, we assume $S = \{1, 2, \ldots, 2t + 2\}$. We will first describe the primitive tools we used in the proposed CRT-based threshold DSS scheme. Note that for each primitive, S is the set of participants for that primitive.

5.1. Joint Random Secret Sharing

In JOINT-RSS, each user in the signing coalition S contributes to the share generation process and obtains a share for the resulting random secret as described below. A verifiable version of this scheme can be found in [13].

1. Each user $j \in S$ chooses a random secret $d_j \in \mathbb{Z}_{m_0}$ and shares it as $d_j \overset{t}{\leftarrow} (y_{1j}, y_{2j}, \cdots, y_{(2t+2)j})$, where y_{ij} is the share of the ith user.
2. The ith user computes $Y_i = \left(\sum_{j=1}^{2t+2} y_{ij}\right) \bmod m_i$. By Proposition 1, $D \overset{t}{\leftarrow} (Y_1, Y_2, \ldots, Y_{2t+2})$, i.e., D can be obtained by using t of $Y_1, Y_2, \ldots, Y_{2t+2}$. Since the Asmuth-Bloom SSS is perfect, D is t-out-of-$(2t + 2)$ secure where $D = \left(\sum_{i=1}^{2t+2} d_i\right) \bmod m_0$, i.e., $t - 1$ users cannot obtain any information on D.

5.2. Joint Zero Sharing

In a JOINT-ZS scheme, each user in the signing coalition S contributes to the zero sharing process and obtains a share for the resulting zero, as described below:

1. Each user $j \in S$ shares $0 \overset{2t}{\leftarrow} (y_{1j}, y_{2j}, \cdots, y_{(2t+2)j})$ by using a $(2t, n)$ Asmuth-Bloom SSS, where $y_{ij} = A_j m_0 \bmod m_i$ is the share of the ith user for some $A_j m_0 < M$. As described in Section 4, due to Lemma 1, the sequence $m_0 < m_1 < \ldots < m_n$ can be used by changing $M = \left\lfloor \frac{\prod_{i=1}^{2t} m_i}{n} \right\rfloor$ instead of (3).

2. The ith user computes $Y_i = \left(\sum_{j=1}^{2t+2} y_{ij}\right) \bmod m_i$. By Proposition 1, $0 \overset{2t}{\leftarrow} (Y_1, Y_2, \ldots, Y_{2t+2})$.

5.3. Computing $g^d \bmod p$

For threshold DSS, we need to share and compute $g^d \bmod p$ for a jointly shared secret $d \in \mathbb{Z}_q$. The scheme Joint-Exp-RSS described below, constructs an intermediate value for $F_d = g^d \bmod p$. This intermediate value will later be corrected through a separate correction process.

1. To compute $F_d = g^d \bmod p$ for a jointly shared secret d, S uses Joint-RSS to generate and share d as $d \overset{t}{\leftarrow} (y_1, y_2, \ldots, y_{2t+2})$, with $m_0 = q$.
2. Each user $i \in S$ computes
$$u_{i,d} = (y_i M_{S\setminus\{i\}} M'_{S,i}) \bmod M_S,$$
where $M'_{S,i} = {M_{S\setminus\{i\}}}^{-1} \bmod m_i$, and broadcasts $f_{i,d} = g^{u_{i,d}} \bmod p$.
3. The intermediate value $F_{d'}$ of $g^d \bmod p$ is computed as $\prod_{i\in S} f_{i,d} \bmod p$.

Observe that $d = ((\sum_{i\in S} u_i) \bmod M_S) \bmod q$, whereas this construction process computes $F_{d'} = g^{d'} \bmod p$ for $d' = \sum_{i\in S} u_i \bmod q$. Since there are $2t+2$ users in S and $u_i < M_S$ for all i, $d = d' - \delta_d M_S \bmod q$ for some integer $0 \le \delta_d \le 2t+1$.

5.4. Computing $g^{k^{-1}} \bmod p$

In DSS, we need to compute $r = g^{k^{-1}} \bmod p$ in such a way that neither k nor k^{-1} is known by any user:

1. S uses Joint-RSS to jointly share random secrets $k \overset{t}{\leftarrow} (k_1, k_2, \ldots, k_{2t+2})$, $a \overset{t}{\leftarrow} (a_1, a_2, \ldots, a_{2t+2})$, and uses Joint-ZS to distribute shares for zero, i.e., $0 \overset{2t}{\leftarrow} (z_1, z_2, \ldots, z_{2t+2})$.
2. S constructs $v = (ak) \bmod m_0$ from shares $v_i = (a_i k_i + z_i) \bmod m_i$, $i \in S$. Note that $v \overset{2t+1}{\leftarrow} (v_1, v_2, \ldots, v_{2t+2})$, as described in Section 4.1.
3. S uses Joint-Exp-RSS to obtain
$$F_{a'} = \prod_{i\in S} f_{i,a} = \prod_{i\in S} g^{u_{i,a}} \equiv g^{a'} \equiv g^{a+\delta_a M_S} \bmod p \text{ and}$$
$$F_{k'} = \prod_{i\in S} f_{i,k} = \prod_{i\in S} g^{u_{i,k}} \equiv g^{k'} \equiv g^{k+\delta_k M_S} \bmod p.$$

S also computes

$$\begin{aligned} F_{a'k'} &= \prod_{i\in S} f_{i,ak} = \prod_{i \in S} {F_{a'}}^{u_{i,k}} \equiv g^{a'k'} \equiv g^{(a+\delta_a M_S)(k+\delta_k M_S)} \bmod p \\ &\equiv g^v {F_{a'}}^{\delta_k M_S} {F_{k'}}^{\delta_a M_S} g^{-\delta_a \delta_k M_S^2} \bmod p. \end{aligned}$$

4. S checks the following equality
$$F_{a'k'} \stackrel{?}{=} g^v {F_{a'}}^{j_k M_S} {F_{k'}}^{j_a M_S} g^{-j_a j_k M_S^2} \bmod p \tag{4}$$
for all $0 \le j_a, j_k \le 2t+1$ and finds the $(j_a = \delta_a, j_k = \delta_k)$ pair that satisfies this equality. Once δ_a is found, $F_a = g^a \bmod p = F_{a'} g^{-\delta_a M_S} \bmod p$ can be computed.
5. The signing coalition S computes $g^{k^{-1}} \bmod p = {F_a}^{(v^{-1})} \bmod p$.

Step 4 of Joint-Exp-Inverse computes $F_a = g^a \bmod p$ from the intermediate value $F_{a'}$. Note that the (j_a, j_k) pair, $0 \le j_a, j_k \le 2t+1$, found for (4) is unique with overwhelming probability, given that $(2t+2)^2 \ll q$.

5.5. Threshold DSS Scheme

The phases of the proposed threshold DSS scheme are described in Fig. 1:

- *Key Generation Phase:* Let $\alpha \in_R \mathbb{Z}_q^*$ be the private signature key. The dealer sets $m_0 = q$ and shares $\alpha \xleftarrow{t} (\alpha_1, \alpha_2, \ldots, \alpha_n)$.
- *Signing Phase:* To sign a hashed message $w \in \mathbb{Z}_q$, the signing coalition S of size $2t+1$ first computes
$$r = (g^{k^{-1}} \bmod p) \bmod q$$
by, Joint-Exp-Inverse described in Section 5.4. To compute $s = k(w + r\alpha) \bmod q$, each user $i \in S$ computes $s_i = (k_i(w + r\alpha_i) + z_i) \bmod m_i$ and broadcasts it where the shares $0 \xleftarrow{2t+1} (z_1, z_2, \ldots, z_{2t+2})$ are obtained by using the primitive Joint-ZS. Then the signature s is computed by using the reconstruction process for the Asmuth-Bloom SSS with $2t+2$ shares.
- *Verification Phase* is the same as the standard DSS verification.

Figure 1: CRT-based threshold DSS signature.

Let $Y^{(\alpha)}$, $Y^{(k)}$, and $Y^{(z)}$ be the smallest integers, such that

$$\begin{aligned} \alpha &= Y^{(\alpha)} \bmod m_0, & k &= Y^{(k)} \bmod m_0, & 0 &= Y^{(z)} \bmod m_0, \\ \alpha_i &= Y^{(\alpha)} \bmod m_i, & k_i &= Y^{(k)} \bmod m_i \text{ and} & z_i &= Y^{(z)} \bmod m_i, \text{ for } i \in S. \end{aligned}$$

Since α and k are jointly shared with threshold t, due to Proposition 1, they can be constructed with t shares. Hence, both $Y^{(\alpha)}$ and $Y^{(k)}$ are less than $\prod_{i=1}^{t} m_i$. On the other hand, $Y^{(z)}$ requires $2t+1$ of the shares $z_1, \ldots, z_{2t+2}$, hence, $Y^{(z)} < \prod_{i=1}^{2t+1} m_i$. Since $w, r < m_0$, we have

$$Y^{(k)} \left(w + rY^{(\alpha)}\right) + Y^{(z)} < m_0 \prod_{i=1}^{t} m_i \left(\prod_{i=1}^{t} m_i + 1\right) + \prod_{i=1}^{2t+1} m_i,$$

which can be computed by a coalition of size $2t+2$ by Proposition 2 and Section 4.1. Hence, $s \overset{2t+2}{\longleftarrow} (s_1, s_2, \ldots, s_n)$, i.e., s can be computed by $2t+2$ partial signatures s_i, $i \in S$.

6. Security Analysis

Here we will prove that the proposed threshold DSS signature scheme is secure (i.e., existentially non-forgeable against an adaptive chosen message attack), provided that the DSS signatures are unforgeable. Throughout the paper, we assume a *static adversary model* where the adversary controls exactly $t-1$ users and chooses them at the beginning of the attack. In this model, the adversary obtains all secret information of the corrupted users and the public parameters of the cryptosystem. She can control the actions of the corrupted users, ask for partial signatures of the messages of her choice, but she is the static in the sense that she cannot corrupt another user in the course of an attack.

For the proof, we will follow the standard methodology for threshold signatures [9, 17]: To reduce the problem of breaking the DSS signature scheme to breaking the proposed threshold DSS scheme, we will simulate the protocol with no information on the secret where the output of the simulator is indistinguishable from an actual run of the protocol from the adversary's point of view. Therefore, an attacker on the threshold DSS will imply an attacker on DSS itself through the simulator. The input to the simulator is the hashed message w, its signature (r, s), the public key β, and the secret shares of the corrupted users, i.e., $\alpha_i \in S_B$, where S_B denotes the corrupted (bad) user set. Let S_G be the set of good users in S and let $r^* = g^{ws^{-1}} \beta^{rs^{-1}} \bmod p$. The simulator is given in Fig. 2.

To prove that the outcome of the simulator is indistinguishable, we first need to state the following assumption:

1. By simulating the good users in S_G, with the JOINT-RSS procedure, the simulator shares random values for each user in S_G. It also obtains the good users' shares from the corrupted users in S_B. Note that all of these values are known by the simulator because $|S_G| \geq t$, which is the threshold. Let $\overline{a}, \overline{k} \in \mathbb{Z}_q$ be the shared values in this step, i.e., $\overline{k} \overset{t}{\leftarrow} (\overline{k}_1, \overline{k}_2, \ldots, \overline{k}_{2t+2})$ and $\overline{a} \overset{t}{\leftarrow} (\overline{a}_1, \overline{a}_2, \ldots, \overline{a}_{2t+2})$. After that, 0 is shared by using the procedure JOINT-ZS, i.e., $0 \overset{2t}{\leftarrow} (\overline{z}_1, \overline{z}_2, \ldots, \overline{z}_{2t+2})$. For the rest of the simulation, let $\delta_{\overline{a}} = \left\lfloor \frac{\sum_{i \in S} u_{i,\overline{a}}}{M_S} \right\rfloor$ and $\delta_{\overline{k}} = \left\lfloor \frac{\sum_{i \in S} u_{i,\overline{k}}}{M_S} \right\rfloor$.
2. By using the second step of JOINT-EXP-INVERSE , $\overline{v} = \overline{a}\overline{k}$ is computed. Let $\overline{F_{a'}} = r^{*^{\overline{v}}} g^{\delta_{\overline{a}} M_S}$. The simulator uses $\overline{a}_i$ values to compute $\overline{f_{i,a}} = g^{\overline{a}_i M_{S \setminus i} M'_{S,i} \bmod M_S} \bmod p$ for all $i \in S_G$ but one. For the last user, $\overline{f_{i,a}}$ is selected such that

$$\prod_{i \in S_G} \overline{f_{i,a}} \equiv \overline{F_{a'}} \left(\prod_{i \in S_B} \overline{f_{i,a}} \right)^{-1} \bmod p. \tag{5}$$

   These $\overline{f_{i,a}}$ values are then broadcast.
3. By using the construction phase of JOINT-EXP-RSS, $F_{\overline{k'}} = \prod_{i \in S} f_{i,\overline{k}} \bmod p$ is computed. Let

$$\overline{F_{a'k'}} = \overline{F'_a}^{\delta_{\overline{k}} M_S} F_{\overline{k'}}^{\delta_{\overline{a}} M_S} g^{-\delta_{\overline{a}} \delta_{\overline{k}} M_S^2} g^{\overline{v}} \bmod p.$$

   The simulator uses $\overline{k}_i$ values to compute $\overline{f_{i,ak}} = \overline{F_{a'}}^{\overline{k_i} M_{S \setminus i} M'_{S,i} \bmod M_S} \bmod p$ for all $i \in S_G$ but one. For the last user, $\overline{f_{i,ak}}$ is selected such that

$$\prod_{i \in S_G} \overline{f_{i,ak}} \equiv \overline{F_{a'k'}} \left(\prod_{i \in S_B} \overline{f_{i,ak}} \right)^{-1} \bmod p.$$

   These $\overline{f_{i,ak}}$ values are then broadcasted and after that the correction phase is completed.
4. Let $\overline{s_i} = (\overline{k}_i (w + \alpha_i r) + z_i) \bmod q$ for $i \in S_B$, where z_is are obtained from the execution of a JOINT-ZS with threshold $2t+1$. The simulator chooses a random integer $\overline{U_s}$ smaller than

$$m_0 \prod_{i=1}^{t} m_i \left(\prod_{i=1}^{t} m_i + 1 \right) + \prod_{i=1}^{2t+1} m_i,$$

   such that $\overline{U_s} \equiv \overline{s_i} \bmod m_i$ for $i \in S_B$ and $\overline{U_s} \equiv s \bmod m_0$. Then it computes $\overline{s_i} = \overline{U_s} \bmod m_i$ for $i \in S_G$ and broadcasts them. After these steps, the signature (r, s) is computed.

Figure 2: Simulator for the threshold DSS protocol.

Assumption 1 (Gennaro et al. [8]). *Let G be the subgroup generated by g. Choose u, v at random, uniformly distributed and independently, in $\mathbb{Z}_q$. The following probability distributions on $G \times G$, $(g^u \bmod p, g^v \bmod p)$ and*

$\left(g^u \bmod p, g^{u^{-1}} \bmod p\right)$, *are computationally indistinguishable.*

In a related work, Bao et al. [2] prove that if the Computational Diffie-Hellman (CDH) assumption holds, there is no polynomial-time algorithm that outputs $g^{x^{-1}}$ on inputs g and g^x with non-negligible probability. We use this assumption in Lemma 2 to prove the security of the overall scheme:

Lemma 2. *The outcome of the simulator in Fig.2 is indistinguishable from the CRT-based threshold DSS scheme from a static adversary's point of view under Assumption 1 and the DDH assumption.*

PROOF. 1. As shown by Theorem 1, the modified SSS is perfect, i.e., the probabilities $\Pr(d = \overline{k})$ and $\Pr(d = k)$ are equal for $k, \overline{k} \in \mathbb{Z}_{m_0}$, where d is the shared secret, k is the shared value in real protocol, and $\overline{k}$ is the shared value in the simulation. The same argument is also true for $\overline{a}$.

2. In the real protocol, the set of shares $(v_1, v_2, \ldots, v_{2t+2})$ is a valid sharing for a uniformly distributed value v. In the simulation, $(\overline{v}_1, \overline{v}_2, \ldots, \overline{v}_{2t+2})$ also yields a uniformly distributed value $\overline{v}$. Hence, the distribution of the shares v_i, $i \in S$, is identical to the distribution of $\overline{v}_i$, $i \in S$. Note that, if the joint zero-sharing procedure is not used, i.e., if the shares of v are not randomized, the secrecy of a and k is not preserved.
In the real protocol, $F_{a'} = g^{a+\delta_a M_S} \bmod p$, where a is random and δ_a is another random value independent from a. The simulation computes $\overline{F_{a'}} = g^{k^{-1}\overline{v}+\delta_{\overline{a}} M_S} \bmod p$. Since $\overline{v}$ is uniformly random, $k^{-1}\overline{v}$ is also uniformly random. The simulator uses the exact $\delta_{\overline{a}}$ value determined in Step 2 so its distribution is identical to that of δ_a. Hence, the distribution of $F_{a'}$ and $\overline{F_{a'}}$ values are identical.
In the simulation, $\overline{a}_i$s are used to compute $\overline{f_{i,a}} = g^{\overline{a}_i M_{S\setminus i} M'_{S,i} \bmod M_S} \bmod p$, and in computing of $f_{i,a} = g^{u_{i,a}}$, thanks to perfectness, the share a_i can be any integer from $\mathbb{Z}_q$. Hence, the distributions of $f_{i,a}$s and $\overline{f_{i,a}}$s are indistinguishable for the users in $S \setminus \{j\}$. However, for the last user j of S_G, the simulator chooses a specific $\overline{f_{j,a}}$ to satisfy (5). We will show that without $\overline{f_{j,a}}$, the rest of the $\overline{f_{i,a}}$s for $i \in S \setminus \{j\}$ yield a random value in the group generated by g. Let $S' = S \setminus \{j\}$. Consider

$$\sum_{i \in S'} u_{i,\overline{a}} = \sum_{i \in S'} \overline{a}_i M_{S\setminus\{i\}} M'_{S,i} \bmod M_S.$$

Since the threshold for $\overline{a}$ is t and $|S'| > t$, the following equation is satisfied:

$$\overline{a} = \left(\left(\sum_{i \in S'} u_{i,\overline{a}}\right) \bmod M_{S'}\right) \bmod q.$$

However, when we try to do the same construction in the exponent,

$$\prod_{i \in S'} \overline{f_{i,a}} \equiv g^{\overline{a} + \Delta_{\overline{a}} M_{S'}} \bmod p$$

for some $\Delta_{\overline{a}} < |S'| \frac{M_S}{M_{S'}} = |S'| m_j$. Since $\Delta_{\overline{a}}$ is an unknown, $m_j > {m_0}^2 = q^2$, and $\gcd(q, M_{S'}) = 1$, we have $g^{\overline{a} + \Delta_{\overline{a}} M_{S'}}$ uniformly random in the group generated by g. Note that this is true for every $S' \subset S$, i.e., there is no correlation between $\overline{f_{i,a}}$s for $i \in S'$ when $u_{i,\overline{a}}$s are computed for a larger coalition $S \supset S'$. Hence, the distributions of $\overline{f_{i,a}}$ and $f_{i,a}$ for $i \in S$ are indistinguishable.
The same argument is also true for the distributions of $\overline{f_{i,ak}}$ and $f_{i,ak}$ for $i \in S$, which are used in the following step.

3. For the correction phase in the real protocol, the values $f_{i,ak}$ and $f_{i,k}$ use the same value $u_{i,k}$ in the exponent, likewise, the ones in the simulator. The correction equation used in the real protocol is

$$F_{a'k'} = {F_{a'}}^{\delta_k M_S} {F_{k'}}^{\delta_a M_S} g^{-\delta_a \delta_k M_S^2} g^v \bmod p.$$

And, the simulator uses the value

$$\overline{F_{a'k'}} = \overline{F_{a'}}^{\delta_{\overline{k}} M_S} F_{\overline{k'}}^{\delta_{\overline{a}} M_S} g^{-\delta_{\overline{a}} \delta_{\overline{k}} M_S^2} g^{\overline{v}} \bmod p,$$

where the distribution of each value on the right side is identical to that of the corresponding value in the real protocol. Hence, the distribution of $F_{a'k'}$ and $\overline{F_{a'k'}}$ values are identical. The distributions of the outputs of the correction processes, i.e., δ_a and δ_k, are also identical since the simulator uses the actual $\delta_{\overline{a}}$ and $\delta_{\overline{k}}$ values.
Here we need to use the DDH assumption and Assumption 1. After the correction, $(\overline{F_{a'}}, F_{\overline{k'}}, \overline{F_{a'k'}})$ are revealed, where, in the real protocol they are equal to $(g^{a'}, g^{k'}, g^{a'k'})$. The DDH assumption states that the distributions of these two triplets are indistinguishable. Besides, $F_{\overline{k'}} = g^{\overline{k'}}$ and, once $\delta_{\overline{k}}$ is found, $g^{\overline{k}}$ can be computed. The users will also know $r = g^{k^{-1}}$, and in the real protocol the pair $(g^{\overline{k}}, g^{k^{-1}})$ will be $(g^k, g^{k^{-1}})$. Assumption 1 says that the distributions of these two pairs are also indistinguishable.

4. In the real protocol, the set of shares $(s_1, s_2, \ldots, s_{2t+2})$ is a valid sharing for a uniformly distributed value s. In the simulation, $(\overline{s}_1, \overline{s}_2, \ldots, \overline{s}_{2t+2})$ also yields the same value. The computing process of $\overline{s}_i$ for $i \in S_G$ is the same as the one in the real protocol. Hence, the distribution of the shares s_i, $i \in S$, is identical to the distribution of $\overline{s}_i$, $i \in S$. $\square$

We conclude this section with a corollary of Theorem 1 and Lemma 2.

Corollary 1. *Given that the standard DSS scheme is secure, the threshold DSS signature scheme is also secure under the static adversary model.*

7. Information Efficiency of the Proposed Scheme

An important criterion for the efficiency of a threshold scheme is the *information rate*, i.e., the ratio of the secret size to the maximum share size. In our scheme, the domain for the secret is $\mathbb{Z}_{m_0}$ and the domains for the shares are $\mathbb{Z}_{m_i}$ for $1 \leq i \leq n$. Let $|m|$ denote the bit size of an integer m. Since m_n is the largest modulus, the maximum share size is $|m_n|$ and the information rate of the scheme is $|m_0|/|m_n|$. Note that, since $m_1, \ldots, m_n$ are selected as consecutive primes that satisfy (2), each m_i must be greater than nm_0^2.

The prime number theorem says that the density of primes less than x is $1/\ln x$. Let m_1 be the first prime after nm_0^2. Considering that the density of the primes around nm_0^2 is $1/\ln(nm_0^2)$,

$$m_n \approx nm_0^2 + (n-1)\ln(nm_0^2) < n(m_0^2 + 2\ln m_0 + \ln n).$$

Therefore, the information rate of the scheme is

$$\frac{|m_0|}{|n| + |m_0^2 + 2\ln m_0 + \ln n|},$$

and considering $m_0 \gg n$, this number is close to 1/2.

8. Computation Efficiency of the Proposed Scheme

To evaluate the computation efficiency of the proposed scheme, we will count the number of inversions and exponentiations involved: First note that the Asmuth-Bloom SSS uses only multiplication, addition, and modular reduction in the dealing phase, hence in the JOINT-RSS protocol the users do not use any inversion or exponentiation. The same is also true for the JOINT-ZS phase of the proposed protocol. To compute $g^d \mod p$ for a shared secret d, i.e., in the JOINT-EXP-RSS protocol, each user in the coalition performs an inversion to compute $u_{i,d} = (y_i M_{S\setminus\{i\}} M'_{S,i}) \bmod M_S$ and an exponentiation to compute $f_{i,d} = g^{u_{i,d}} \bmod p$.

The proposed protocol uses JOINT-EXP-INVERSE to obtain r, which performs three JOINT-EXP-RSSes, and each user performs three exponentiations with one common inversion. In addition, for equation (4) (step 4 in Section 5.4), the coalition needs to perform four exponentiations and one inversion (and $\mathcal{O}(t^2)$ multiplications). In the final step, we need one more inversion and exponentiation. Hence, in total, each user in the coalition performs three exponentiations and one inversion; and the coalition needs to perform two more inversions and five more exponentiations together.

An exact computational comparison with the Shamir-based threshold DSS scheme [9] is hard to achieve since the characteristics of the arithmetic operations are different. For example, the secret sharing phase in Shamir SSS uses $\mathcal{O}(t)$ exponentiations as opposed to none in ours, but the exponents are small, i.e., less than t. The cost of the JOINT-RSS, JOINT-ZS, JOINT-EXP-RSS protocols are roughly the same, except for these exponentiations in Shamir SSS. The main additional cost of the new protocol is the correction procedure (4). But, as it can be seen from the discussion above, the marginal cost of this step does not dominate the overall computation cost. Furthermore, (4) needs to be performed only once by the whole coalition, and the candidate (j_a, j_k) pairs can be distributed among the users in parallel, where the final signature is verified by each user independently. Therefore, we can say that the computational cost of the proposed scheme is comparable to that of [9], with the exact performance figures depending on the details of the implementations.

9. Conclusion

In this paper, we investigated how the DSS signature function can be shared using the Chinese Remainder Theorem. We proposed a threshold DSS signature scheme based on Asmuth-Bloom secret sharing. The proposed scheme is secure against an adversary who is allowed to corrupt $t - 1$ users, and $2t + 2$ users are required to generate a DSS signature. Although it is slightly less efficient than the Shamir-based solution of Gennaro et al. [9], this is rather normal considering the vast amount of literature on Shamir-based

FSSes. We believe that CRT-based FSSes will keep improving, but more work is needed to have them as efficient as their Shamir-based counterparts.

References

[1] C. Asmuth and J. Bloom. A modular approach to key safeguarding. *IEEE Trans. Information Theory*, 29(2):208–210, 1983.

[2] F. Bao, R. H. Deng, and H. Zhu. Variations of Diffie-Hellman assumption. In *ICICS 2003*, volume 2836 of *LNCS*, pages 301–312.

[3] G. Blakley. Safeguarding cryptographic keys. In *AFIPS National Computer Conference*, 1979.

[4] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *CRYPTO'89*, volume 435 of *LNCS*, pages 307–315. Springer-Verlag, 1990.

[5] Y. Desmedt and Y. Frankel. Shared generation of authenticators and signatures. In *CRYPTO'91*, volume 576 of *LNCS*, pages 457–469.

[6] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Information Theory*, 31(4):469–472, 1985.

[7] National Institute for Standards and Technology. Digital signature standard (DSS). Technical Report 169, August 30, 1991.

[8] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust Threshold DSS Signatures. In *EUROCRYPT 1996*, volume 1070 of *LNCS*, pages 354–371.

[9] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust threshold DSS signatures. *Information and Computation*, 164(1):54–84, 2001.

[10] S. Iftene, S. Ciobaca, and M. Grindei. Compartmented threshold RSA based on the CRT. ePrint Archive, 2008/370.

[11] S. Iftene and M. Grindei. Weighted threshold RSA based on the CRT. In *SYNACS'2007*, pages 175–181.

[12] K. Kaya and A. A. Selcuk. Threshold cryptography based on Asmuth-Bloom secret sharing. *Information Sciences*, 177(19):4148–4160, 2007.

[13] K. Kaya and A. A. Selcuk. A verifiable secret sharing scheme based on the CRT. In *INDOCRYPT 2008*, volume 5365 of *LNCS*, pages 414–425.

[14] M. Quisquater, B. Preneel, and J. Vandewalle. On the security of the threshold scheme based on the CRT. In *PKC'02*, volume 2274 of *LNCS*, pages 199–210.

[15] A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely? In *STOC'94*, pages 522–533. ACM, 1994.

[16] A. Shamir. How to share a secret? *Comm. ACM*, 22(11):612–613, 1979.

[17] V. Shoup. Practical threshold signatures. In *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 207–220.