Sharing DSS by the Chinese Remainder Theorem

Kamer Kaya, Ali Aydın Selçuk

Department of Computer Engineering
Bilkent University
Ankara, 06800, Turkey
{kamer,selcuk}@cs.bilkent.edu.tr

November 2, 2010

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 the CRT.

Keywords: 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 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 FSSs 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 [11] showed how to use the Asmuth-Bloom SSS [1] for function sharing. In that paper [11], 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 neat solution was given by Gennaro et al. [8], 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. [8] 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 of [11] to compute the correct function output values from the partial results. The proposed scheme is provably secure 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).

The paper is organized as follows: In Section 2, we give an overview of DSS. We discuss the Asmuth-Bloom SSS in detail in Section 3 and our modifications on the basic scheme in Section 4. We describe our threshold DSS scheme in Section 5, prove its security in Section 6, and discuss its efficiency in Section 7. We conclude the paper in Section 8.

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

$$r \stackrel{?}{=} (g^{ws^{-1}} \beta^{rs^{-1}} \bmod p) \bmod q,$$

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 by modular arithmetic such that any t users can reconstruct the secret by the CRT. The scheme presented below is a slightly modified version, by Kaya and Selcuk [11], 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
  $$\prod_{i=1}^{t} m_i > {m_0}^2 \prod_{i=1}^{t-1} m_{n-i+1}. \quad (1)$$
  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 subject to the condition that $0 \leq y < M$.
  3. The share of the ith user, $1 \leq i \leq n$, 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
  $$d = y \bmod m_0.$$

According to the Chinese Remainder Theorem, y can be determined uniquely in $\mathbb{Z}_{M_S}$ since $y < M \leq M_S$ for any coalition S of size t.

It is shown in [11] that the Asmuth-Bloom version presented here is *perfect* in the sense that no coalition of size smaller than t can obtain any information about the secret:

Theorem 1 (Kaya and Selcuk [11]) *For a passive adversary with $t-1$ shares, every integer in $\mathbb{Z}_{m_0}$ is a secret candidate. Furthermore, every candidate for the secret is equally likely, i.e., the probabilities $\Pr(d = d')$ and $\Pr(d = d'')$ are equal for all $d', d'' \in \mathbb{Z}_{m_0}$.*

Proof 2 *Suppose the adversary corrupts $t-1$ users and just observes the inputs and outputs of the corrupted users without controlling their actions, i.e., the adversary is honest in user actions but curious about the secret. Let S' be the adversarial coalition of size $t-1$, and let y' be the unique solution for y in $Z_{M_{S'}}$. According to (1), $M/M_{S'} > m_0$, hence $y'+jM_{S'}$ is smaller than M for $j < m_0$. Since $\gcd(m_0, M_{S'}) = 1$, all $(y' + jM_{S'}) \bmod m_0$ are distinct for $0 \leq j < m_0$, and there are m_0 of them. That is, d can be any integer from $\mathbb{Z}_{m_0}$.*

For each value of d, there are either $\lfloor M/(M_{S'}m_0) \rfloor$ or $\lfloor M/(M_{S'}m_0) \rfloor + 1$ possible values of y consistent with d, depending on the value of d. Note that $M/(M_{S'}m_0) > m_0 \gg 1$, Hence, for two different integers in $\mathbb{Z}_{m_0}$, the probabilities of d equals these integers are same and hence, d values are equally likely.

Quisquater et al. [14] showed that when moduli m_i are chosen as consecutive primes $1 \leq i \leq n$, the scheme has better security properties. In this paper, we assume that all m_i are selected as consecutive primes. We also assume that $m_0 \gg n$ and $n \geq 2t + 2$.

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 3.

Lemma 3 *An $(\lceil n/2 \rceil, n)$ Asmuth-Bloom sequence is a (k, n) Asmuth-Bloom sequence for all k such that $1 \leq k \leq n$.*

Proof 4 *Let (1) be satisfied with $t = \lceil n/2 \rceil$. Let $k < t$. Rewriting (1) as*

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

and observing $m_i \leq m_{n-i+2}$ *for* $k+1 \leq i \leq t$,

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

follows. Now let $t = \lceil n/2 \rceil$ *and* $k > t$. *Since (1) is satisfied and* $m_i \geq m_{n-i+2}$ *for* $t+1 \leq i \leq k$,

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

and hence, (2) holds as well.

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

$$\prod_{i=1}^{t} m_i > n{m_0}^2 \prod_{i=1}^{t-1} m_{n-i+1}. \tag{3}$$

Note that Lemma 3 is also valid for this equation. Lastly, we change the definition of M to

$$M = \left\lfloor \frac{\prod_{i=1}^{t} m_i}{n} \right\rfloor. \tag{4}$$

The last two modifications, (3) and (4), were proposed by Kaya and Selcuk [12] to make secure joint random secret sharing (Joint-Rss) practical, which is a primitive we use for the proposed threshold DSS scheme. On the other hand, using an $(\lceil n/2 \rceil, n)$ Asmuth-Bloom sequence will be useful for the joint zero sharing primitive (Joint-ZS), in which we require a $(2t, n)$ sequence with the same m_i, $1 \leq i \leq n$, as used in the other primitives where the threshold is t.

This modified scheme was proven secure in [12], with a proof similar to that of Theorem 1:

Theorem 5 (Kaya and Selcuk [12]) *The modified secret sharing scheme with the new* M *and Eq. (3) is perfect in the sense that the probabilities* $\Pr(d = d')$ *and* $\Pr(d = d'')$ *are equal for all* $d', d'' \in \mathbb{Z}_{m_0}$.

4.1 Arithmetic Properties of the Modified Asmuth-Bloom SSS

Suppose several secrets are shared with common parameters t, n, and moduli m_i for $1 \leq i \leq n$, chosen according to (3). The shareholders can use the following properties to obtain new shares for the sum and product of the shared secrets.

Proposition 6 *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.*

Proof 7 *For $Y = \sum_{i=1}^{n}(d_i + A_i m_0)$, we have $Y_i \equiv Y \bmod m_i$. Note that due to (4), $Y < nM < M_S$ for any coalition S where $|S| \geq t$. Hence, a coalition S of t users can construct $Y \in M_S$ and obtain $D = Y \bmod m_0$.*

Note that if the modification in (4) is not applied, the shares $(Y_1, Y_2, \cdots, Y_n)$ can still be used to construct $D = (\sum_{i=1}^{n} d_i) \bmod m_0$. However, in this case, instead of t, we require at least $t+1$ of them to obtain the smallest integer Y that satisfies all the congruences $Y \equiv Y_i \pmod{m_i}$ for $1 \leq i \leq n$. Note that $Y < nM$ since we have n secrets d_i, $1 \leq i \leq n$, in the summation, and for each d_i, the corresponding y used in the dealing phase of the Asmuth-Bloom SSS is smaller than M. Hence, if a coalition S wants to obtain Y, it requires that $M_S \geq nM$ and this is true for all coalitions S of cardinality $t+1$ if (4) is not applied. As next proposition shows, such an increase still occurs when we consider the product of two secrets.

Proposition 8 *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 $D \overset{2t}{\leftarrow} (Y_1, Y_2, \cdots, Y_n)$.*

Proof 9 *For $Y = \prod_{i=1}^{2}(d_i + A_i m_0)$, we have $Y_i \equiv Y \bmod m_i$. Note that $Y < M^2 < M_S$ for any coalition S where $|S| \geq 2t$. Hence, a coalition S of $2t$ users can construct $Y \in M_S$ and obtain $D = Y \bmod m_0$, since $Y = D + Am_0$ for $A = A_1A_2 + A_1d_2 + A_2d_1$.*

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 in a threshold fashion 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.

Note that the secret α is shared with a (t, n) Asmuth-Bloom SSS. Similar to the underlying secret sharing scheme, the proposed scheme is secure against a static adversary, who can corrupt at most $t-1$ users. On the other hand, the required number of users in a signing coalition is $2t+2$. 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 a JOINT-RSS scheme, each user in the signing coalition S contributes something 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 [12].

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 6, $D \overset{t}{\leftarrow} (Y_1, Y_2, \ldots, Y_{2t+2})$, 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 something 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 3, 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 (4).

2. The ith user computes

$$Y_i = \left(\sum_{j=1}^{2t+2} y_{ij} \right) \bmod m_i.$$

By Proposition 6, $0 \overset{2t}{\leftarrow} (Y_1, Y_2, \ldots, Y_{2t+2})$.

Even though the shared value is known to be zero in a JOINT-ZS procedure, from an adversary's point of view, each A_j for $j \in S$, and hence $Y = \sum_{j \in S} A_j$, remains unknown. Note that $0 \leq A_j m_0 < M$ and for an adversarial coalition S', there are $M/M_{S'}$ candidates for each $A_j m_0$, $j \in S$. Hence, there are at least

$$\left\lfloor \frac{\prod_{i=1}^{2t} m_i}{n m_0 M_{S'}} \right\rfloor > \prod_{i=1}^{t} m_i \tag{5}$$

candidates for each A_j. Considering $|S'| = t - 1$, an adversary cannot find each A_j and hence Y.

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 $M'_{S,i}$ is the inverse of $M_{S \setminus \{i\}} \bmod m_i$, and broadcasts

$$f_{i,d} = g^{u_{i,d}} \bmod p.$$

3. The intermediate value for $g^d \bmod p$ is computed as

$$F_{d'} = \prod_{i \in S} f_{i,d} \bmod p.$$

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 $0 \leq \delta_d \leq 2t+1$.

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. The JOINT-EXP-INVERSE procedure described below uses the JOINT-RSS, JOINT-ZS, and JOINT-EXP-RSS primitives and computes r without revealing k:

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

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

4. S checks the following equality

$$F_{a'k'} \overset{?}{=} g^v {F_{a'}}^{j_k M_S} {F_{k'}}^{j_a M_S} g^{-j_a j_k M_S^2} \bmod p \tag{6}$$

for all $0 \leq j_a, j_k \leq 2t+1$ 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

$$g^{k^{-1}} \bmod p = {F_a}^{(v^{-1})} \bmod p.$$

In Step 2 of JOINT-EXP-INVERSE, it is stated that $2t+1$ of $v_1, v_2, \ldots, v_{2t+2}$ are required to compute $v = (ak) \bmod m_0$. This is because, instead of constructing a and k and then multiplying them, in step 2, each user computes his share $v_i = (a_i k_i + z_i) \bmod m_i$ for v, and they compute v by using the combining phase of the Asmuth-Bloom SSS given in Section 3. With this approach, neither a or k is revealed, and hence the proposed scheme remains secure. Note that the required number of shares for a and k are both t. Hence, by Proposition 8, $2t$ shares are sufficient to compute ak from $a_i k_i$, $j \in S$. However, such an approach may reveal some information on a and k. Let $Y^{(v)}$ be the smallest integer such that

$$v = Y^{(v)} \bmod m_0 \text{ and}$$
$$a_i k_i = Y^{(v)} \bmod m_i \text{ for } i \in S,$$

which will be revealed in the combining phase. Note that with this approach $Y^{(v)} = Y^{(a)}Y^{(k)}$ and one may obtain some information on $Y^{(a)}$ and $Y^{(k)}$. On the other hand, if the coalition S uses the shares $(a_i k_i + z_i) \bmod m_i$ to compute v, $Y^{(v)}$ will be equal to $Y^{(a)}Y^{(k)} + Y^{(z)}$. Since the number of candidates for $Y^{(z)}$ is equal to that in (5), we avoid this problem. But this time, as described in Section 4.1, the number of required v_i values to compute v increases by one and becomes $2t+1$.

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 (6) 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:

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

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

Since α and k are jointly shared with threshold t, due to Proposition 6, 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

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

- *Key Generation Phase:* Let $\alpha \in_R \mathbb{Z}_q^*$ be the private signature key. The dealer sets $m_0 = q$ and shares $\alpha \overset{t}{\leftarrow} (\alpha_1, \alpha_2, \ldots, \alpha_n)$.
- *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 $0 \overset{2t+1}{\leftarrow} (z_1, z_2, \ldots, z_{2t+2})$ 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.

which can be computed by a coalition of size $2t+2$ by Proposition 8 and Section 4.1. Hence, $s \overset{2t+2}{\leftarrow} (s_1, s_2, \ldots, s_n)$, 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.

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. 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:

Assumption 10 (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 $G \times G$,*

$$(g^u \bmod p, g^v \bmod p) \text{ and } \left(g^u \bmod p, g^{u^{-1}} \bmod p\right),$$

are computationally indistinguishable.

We used this assumption in Lemma 11 to prove the security of the overall scheme. A similar assumption is also used in [13] to prove the security of the proposed anonymous fingerprinting scheme. And in [2], Bao et al. prove that if the Computational Diffie-Hellman (CDH) assumption holds, there is no probabilistic polynomial time Turing machine that outputs $g^{x^{-1}}$ on inputs g and g^x with non-negligible probability. Note that the CDH assumption states that there is no probabilistic polynomial-time Turing machine that outputs g^{xy} on inputs g, g^x, and g^y with non-negligible probability. The decisional version of the CDH is called the Decisional Diffie-Hellman (DDH) assumption. The DDH assumption states that the following probability distributions on $G \times G \times G$,

$$(g^u, g^v, g^{uv}) \text{ and } (g^u, g^v, g^z),$$

are computationally indistinguishable where u, v, z are random, uniformly distributed, and independent in $\mathbb{Z}_q$.

Lemma 11 *The outcome of the simulator in Fig.2 is indistinguishable from the CRT-based threshold DSS signature from a static adversary's point of view.*

Proof 12

1. *As shown in Theorem 5, 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.*

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

$$\prod_{i \in S_G} \overline{f_{i,a}} \equiv \overline{F_{a'}} \left(\prod_{i \in S_B} \overline{f_{i,a}} \right)^{-1} \bmod p. \tag{7}$$

   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.

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 (7). 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 10. 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 10 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 $\overline{s}_i$, $i \in S$.*

We conclude this section with a corollary of Theorem 5 and Lemma 11.

Corollary 13 *Given that the standard DSS signature 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*, defined as

$$\frac{\text{secret size}}{\text{maximum share size}}.$$

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 (3), 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 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.

The resulting scheme turns out to be less efficient than the Shamir-based threshold DSS scheme of Gennaro et al. [8]. This is rather normal considering that traditionally, the Asmuth-Bloom SSS has not been deemed suitable for function sharing, and CRT-based techniques were not used in function sharing applications until the recent works of Kaya and Selcuk [11, 12] and Iftene et al. [9, 10]. We believe that CRT-based function sharing applications will keep improving, but more work is needed to see 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 *Proc. of ICICS 2003*, volume 2836 of *LNCS*, pages 301–312. Springer-Verlag, 2003.

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

[4] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Proc. of 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 *Proc. of CRYPTO'91*, volume 576 of *LNCS*, pages 457–469. Springer-Verlag, 1992.

[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. *Information and Computation*, 164(1):54–84, 2001.

[9] S. Iftene, S. Ciobaca, and M. Grindei. Compartmented threshold RSA based on the Chinese Remainder Theorem. Cryptology ePrint Archive, Report 2008/370, 2008.

[10] S. Iftene and M. Grindei. Weighted threshold RSA based on the Chinese Remainder Theorem. In *Proc. of SYNACS'2007*, pages 175–181. IEEE Computer Society Press, 2007.

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

[12] K. Kaya and A. A. Selcuk. A verifiable secret sharing scheme based on the Chinese Remainder Theorem. In *Proc. of INDOCRYPT 2008*, volume 5365 of *LNCS*, pages 414–425. Springer-Verlag, 2008.

[13] B. Pfitzmann and A. Sadeghi. Anonymous fingerprinting with direct non-repudiation. In *Proc. of ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 401–414. Springer-Verlag, 2000.

[14] M. Quisquater, B. Preneel, and J. Vandewalle. On the security of the threshold scheme based on the Chinese Remainder Theorem. In *PKC'02: Proceedings of the 5th International Workshop on Practice and Theory in Public Key Cryptosystems*, volume 2274 of *LNCS*, pages 199–210. Springer-Verlag, 2002.

[15] A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely? In *Proc. of 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 *Proc. of EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 207–220. Springer-Verlag, 2000.