Sharing DSS by the Chinese Remainder Theorem

Kamer Kaya[1], **Ali Aydın Selçuk**[2]

[1] *Ohio State University, OH, USA* kamer@bmi.osu.edu
[2] *Bilkent University, Turkey* selcuk@cs.bilkent.edu.tr

Extended Abstract

1 Introduction

Threshold cryptography deals with the problem of sharing a sensitive secret among a group of n users so that the secret can be reconstructed only when 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, 2, 6]).

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 traditionally used Shamir's SSS [6] until a recent work by Kaya and Selcuk [5] showed how to use the Asmuth-Bloom SSS [1] for function sharing.

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. [4], based on Shamir's SSS.

In this paper, we propose a new threshold scheme for DSS with the Asmuth-Bloom SSS. To the best of our knowledge, this is the first provably secure threshold DSS scheme based on the Chinese Remainder Theorem (CRT). In this extended abstract, we omit the proofs which will be given in the full version of the paper.

2 Digital Signature Standard

The DSS signature scheme [3] 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 such that any t users can reconstruct the secret by the CRT. The scheme below is a slightly modified version by Kaya and Selcuk [5] in order to obtain better security properties.

- *Dealing Phase:* To share a secret d among n users, 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 $0 \leq y < M$.
  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, and $M_S = \prod_{i \in S} m_i$.

  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 $\mathbb{Z}_{m_i}$. First, the ith user computes

$$u_i = y_i M'_{S,i} M_{S\setminus\{i\}} \bmod M_S.$$

  2. The users compute $y = \sum_{i \in S} u_i \bmod M_S$ 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 $1 \leq k \leq n$.*

To adapt the original scheme for threshold DSS, 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 $M = \prod_{i=1}^{k} m_i$. Then, we multiply the right side of (1) by n and obtain

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

Lastly, we change the definition of M to

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

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 (2). The 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

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 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:

1. Each user $j \in S$ chooses a random secret $d_j \in \mathbb{Z}_{m_0}$ and shares it as $d_j \stackrel{t}{\leftarrow} (y_{1j}, y_{2j}, \cdots, y_{(2t+2)j})$, where y_{ij} is the share of the ith user.

2. The ith user computes $Y_i = \sum_{j=1}^{2t+2} y_{ij} \bmod m_i$. By Proposition 1, $D \stackrel{t}{\leftarrow} (Y_1, Y_2, \ldots, Y_{2t+2})$, for $D = \sum_{i=1}^{2t+2} d_i \bmod m_0$.

In JOINT-ZS, our related zero sharing scheme, each user in S contributes something to the zero sharing process and obtains a share for the resulting zero.

5.2 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 $F_d = g^d \bmod p$, which will later be corrected:

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 $d \stackrel{t}{\leftarrow} (y_1, y_2, \ldots, y_{2t+2})$, 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 $0 \le \delta_d \le 2t+1$.

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

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

$$F_{a'k'} \overset{?}{=} g^v {F_{a'}}^{j_k M_S} {F_{k'}}^{j_a M_S} g^{-j_a j_k M_S^2} \bmod p \tag{4}$$

   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 is computed as,

$$F_a = g^a \bmod p = F_{a'} g^{-\delta_a M_S} \bmod p.$$

5. The signing coalition S computes $g^{k^{-1}} \bmod p = {F_a}^{(v^{-1})} \bmod p$.

5.4 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

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

Since α and k are jointly shared with threshold t, due to Proposition 1, they can be constructed with t shares. Hence, both $Y^{(\alpha)}$ and $Y^{(k)}$ are less than $\prod_{i=1}^{t} m_i$.

- *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 5.3. 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.

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 $2t+2$ by Proposition 2. 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, $i \in S$.

References

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

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

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

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

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

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