

# Robust Threshold Schemes Based on the Chinese Remainder Theorem

Kamer Kaya\* and Ali Aydın Selçuk

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

**Abstract.** Recently, Chinese Remainder Theorem (CRT) based function sharing schemes are proposed in the literature. In this paper, we investigate how a CRT-based threshold scheme can be enhanced with the robustness property. To the best of our knowledge, these are the first robust threshold cryptosystems based on a CRT-based secret sharing.

**Keywords:** Threshold cryptography, robustness, RSA, ElGamal, Paillier, Chinese Remainder Theorem.

## 1 Introduction

In threshold cryptography, *secret sharing* deals with the problem of sharing a highly sensitive secret among a group of  $n$  users so that only when a sufficient number  $t$  of them come together the secret can be reconstructed. *Function sharing* deals with evaluating the encryption/signature function of a cryptosystem without the involved parties disclosing their secret shares. A function sharing scheme (FSS) requires distributing the function's computation according to the underlying secret sharing scheme (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 the individual secrets. Several SSSs [1,3,20] and FSSs [8,9,10,11,19,21] have been proposed in the literature.

Nearly all existing solutions for the function sharing problem have been based on the Shamir SSS [20]. Recently, Kaya and Selçuk [14] proposed several threshold function sharing schemes based on the Asmuth-Bloom SSS for the RSA [18], ElGamal [13] and Paillier [16] cryptosystems. These FSSs are the first examples of secure function sharing schemes based on Asmuth-Bloom secret sharing.

We say that a function sharing scheme is *robust* if it can withstand participation of corrupt users in the function evaluation phase. In a robust FSS, a detection mechanism is used to identify the corrupted partial results so that, the corrupted users can be eliminated. The FSSs proposed by Kaya and Selçuk [14]

---

\* Supported by the Turkish Scientific and Technological Research Agency (TÜBİTAK) Ph.D. scholarship.

did not have the robustness property and, to the best of our knowledge, no CRT-based robust and secure function sharing scheme exists in the literature.

In this paper, we investigate how CRT-based threshold schemes can be enhanced with the robustness property. We first give a robust threshold function sharing scheme for the RSA cryptosystem. Then we apply the ideas to the ElGamal and Paillier decryption functions. For RSA and Paillier, we use the threshold schemes proposed by Kaya and Selçuk [14]. For ElGamal, we work with a modified version of the ElGamal decryption scheme by Wei et al. [22]. All of the proposed schemes are provably secure against a static adversary under the random oracle model [2].

In achieving robustness, we make use of a non-interactive protocol designed to prove equality of discrete logarithms [4,5,21]. The original interactive protocol was proposed by Chaum et al [5] and improved by Chaum and Pedersen [6]. Later, Shoup [21] and, Boudot and Traoré [4] developed a non-interactive version of the protocol.

The organization of the paper is as follows: In Section 2, we describe the Asmuth-Bloom SSS and the FSSs proposed by Kaya and Selçuk [14]. After describing a robust threshold RSA scheme and proving its security in Section 3, we apply the proposed idea to the Paillier and ElGamal cryptosystems in Section 4. Section 5 concludes the paper.

## 2 Function Sharing Based on the Asmuth-Bloom Secret Sharing

The Asmuth-Bloom SSS shares a secret among the parties using modular arithmetic and reconstructs it by the Chinese Remainder Theorem. Here we give the brief description of the scheme:

- *Dealer Phase:* To share a secret  $d$  among a group of  $n$  users with threshold  $t$ , the dealer does the following:
  - A set of pairwise relatively prime integers  $m_0 < m_1 < m_2 < \dots < m_n$  are chosen where  $m_0 > d$  is prime,

$$\prod_{i=1}^t m_i > m_0 \prod_{i=1}^{t-1} m_{n-i+1}. \quad (1)$$

- 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  $0 \leq y < M$ .

- The share of the  $i$ th user,  $1 \leq i \leq n$ , is

$$y_i = y \bmod m_i$$

i.e., the smallest nonnegative residue of  $y$  modulo  $m_i$ .

- *Combiner Phase:* Assume  $\mathcal{S}$  is a coalition of  $t$  users to construct the secret. For any coalition  $\mathcal{S}$ , we define  $M_{\mathcal{S}}$  as

$$M_{\mathcal{S}} = \prod_{i \in \mathcal{S}} m_i.$$

- Given the system

$$y \equiv y_i \pmod{m_i}$$

for  $i \in \mathcal{S}$ , find  $y$  in  $\mathbb{Z}_{M_{\mathcal{S}}}$  using the Chinese Remainder Theorem.

- Compute the secret as

$$d = y \pmod{m_0}.$$

According to the Chinese Remainder Theorem,  $y$  can be determined uniquely in  $\mathbb{Z}_{M_{\mathcal{S}}}$ . Since  $y < M \leq M_{\mathcal{S}}$ , the solution is also unique in  $\mathbb{Z}_M$ .

In the original Asmuth-Bloom scheme,  $m_0$  is not needed until the last step of the combiner phase but still it is a public value. To avoid confusions, we emphasize that it will be secret for the robust FSSs proposed in this paper.

Kaya and Selçuk [14] modified the Asmuth-Bloom SSS by changing (1) as

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

to make the Asmuth-Bloom SSS *perfect* in the sense that  $t - 1$  or fewer shares do not narrow down the key space and furthermore all candidates for the key are equally likely: Assume a coalition  $\mathcal{S}'$  of size  $t - 1$  has gathered and let  $y'$  be the unique solution for  $y$  in  $\mathbb{Z}_{M_{\mathcal{S}'}}$ . According to (2),  $M/M_{\mathcal{S}'} > m_0^2$ , hence  $y' + jM_{\mathcal{S}'}$  is smaller than  $M$  for  $j < m_0^2$ . Since  $\gcd(m_0, M_{\mathcal{S}'}) = 1$ , all  $(y' + jM_{\mathcal{S}'}) \pmod{m_0}$  are distinct for  $0 \leq j < m_0$  hence,  $d$  can be any integer from  $\mathbb{Z}_{m_0}$ . For each value of  $d$ , there are either  $\lfloor M/(M_{\mathcal{S}'}m_0) \rfloor$  or  $\lfloor M/(M_{\mathcal{S}'}m_0) \rfloor + 1$  possible values of  $y$  consistent with  $d$ , depending on the value of  $d$ . Hence, for two different integers in  $\mathbb{Z}_{m_0}$ , the probabilities of  $d$  being equal to these integers are almost equal. Note that  $M/(M_{\mathcal{S}'}m_0) > m_0$  and given that  $m_0 \gg 1$ , all  $d$  values are approximately equally likely.

In the original Asmuth-Bloom SSS, the authors proposed an iterative process to solve the system  $y \equiv y_i \pmod{m_i}$ . Instead, a classical and non-iterative solution exists which is more suitable for function sharing in the sense that it does not require interaction between parties and has an additive structure convenient to share exponentiations [12].

1. Let  $\mathcal{S}$  be a coalition of at least  $t$  users. Let  $M_{\mathcal{S} \setminus \{i\}}$  denote  $\prod_{j \in \mathcal{S}, j \neq i} m_j$  and  $M'_{\mathcal{S},i}$  be the multiplicative inverse of  $M_{\mathcal{S} \setminus \{i\}}$  in  $\mathbb{Z}_{m_i}$ , i.e.,

$$M_{\mathcal{S} \setminus \{i\}} M'_{\mathcal{S},i} \equiv 1 \pmod{m_i}.$$

First, the  $i$ th user computes

$$u_i = (y_i M'_{\mathcal{S},i} \pmod{m_i}) M_{\mathcal{S} \setminus \{i\}}. \tag{3}$$

2.  $y$  is computed as

$$y = \sum_{i \in \mathcal{S}} u_i \text{ mod } M_{\mathcal{S}}.$$

3. The secret  $d$  is computed as

$$d = y \text{ mod } m_0.$$

Even with these modifications, obtaining a threshold scheme by using Asmuth-Bloom SSS is not a straightforward task. Here we give the description of the proposed threshold RSA signature scheme [14].

- *Setup*: In the RSA setup phase, choose the RSA primes  $p = 2p' + 1$  and  $q = 2q' + 1$  where  $p'$  and  $q'$  are also large random primes.  $N = pq$  is computed and the public key  $e$  and private key  $d$  are chosen from  $\mathbb{Z}_{\phi(N)}^*$  where  $ed \equiv 1 \pmod{\phi(N)}$ . Use Asmuth-Bloom SSS for sharing  $d$  with a secret  $m_0 = \phi(N) = 4p'q'$ .
- *Signing*: Let  $w$  be the hashed message to be signed and suppose the range of the hash function is  $\mathbb{Z}_N^*$ . Assume a coalition  $\mathcal{S}$  of size  $t$  wants to obtain the signature  $s = w^d \text{ mod } N$ .

- *Generating the partial results*: Each user  $i \in \mathcal{S}$  computes

$$\begin{aligned} u_i &= (y_i M'_{\mathcal{S},i} \text{ mod } m_i) M_{\mathcal{S} \setminus \{i\}}, \\ s_i &= w^{u_i} \text{ mod } N. \end{aligned} \quad (4)$$

- *Combining the partial results*: The incomplete signature  $\bar{s}$  is obtained by combining the  $s_i$  values

$$\bar{s} = \prod_{i \in \mathcal{S}} s_i \text{ mod } N. \quad (5)$$

- *Correction*: Let  $\kappa = w^{-M_{\mathcal{S}}} \text{ mod } N$  be the *corrector*. The incomplete signature can be corrected by trying

$$(\bar{s}\kappa^j)^e = \bar{s}^e (\kappa^e)^j \stackrel{?}{\equiv} w \pmod{N} \quad (6)$$

for  $0 \leq j < t$ . Then the signature  $s$  is computed by

$$s = \bar{s}\kappa^\delta \text{ mod } N$$

where  $\delta$  denotes the value of  $j$  that satisfies (6).

- *Verification* is the same as the standard RSA verification where the verifier checks

$$s^e \stackrel{?}{\equiv} w \pmod{N}$$

The signature  $\bar{s}$  generated in (5) is *incomplete* since we need to obtain  $y = \sum_{i \in S} u_i \bmod M_S$  as the exponent of  $w$ . Once this is achieved, we have  $w^y \equiv w^d \pmod{N}$  as  $y = d + Am_0$  for some  $A$  where  $m_0 = \phi(N)$ .

Note that the equality in (6) must hold for some  $j \leq t - 1$  since the  $u_i$  values were already reduced modulo  $M_S$ . So, combining  $t$  of them in (5) will give  $d + am_0 + \delta M_S$  in the exponent for some  $\delta \leq t - 1$ . Thus in (5), we obtained

$$\bar{s} = w^{d+\delta M_S} \bmod N = sw^{\delta M_S} \bmod N = s\kappa^{-\delta} \bmod N$$

and for  $j = \delta$ , equation (6) will hold. Also note that the mappings  $w^e \bmod N$  and  $w^d \bmod N$  are bijections in  $\mathbb{Z}_N$ , hence there will be a unique value of  $s = \bar{s}\kappa^j$  which satisfies (6).

Besides RSA, Kaya and Selçuk also applied this *combine-and-correct* approach to obtain threshold Paillier and ElGamal schemes [14] with Asmuth-Bloom secret sharing.

### 3 Robust Sharing of the RSA Function

To enhance the threshold cryptosystems with the robustness property, we use a non-interactive protocol proposed to prove equality of two discrete logarithms with respect to different moduli. The interactive protocol, which was originally proposed by Chaum et al [5] for the same moduli, was modified by Shoup and used to make a threshold RSA signature scheme robust [21]. He used Shamir’s SSS as the underlying SSS to propose a practical and robust threshold RSA signature scheme. In Shamir’s SSS, the secret is reconstructed by using Lagrange’s polynomial evaluation formula and all participants use the same modulus which does not depend on the coalition. On the other hand, in the direct solution used in the abovementioned CRT-based threshold RSA scheme, the definition of  $u_i$ s in (3) and (4) shows that we need different moduli for each user. For robustness, we need to check the correctness of  $u_i$  for each user  $i$  in the function evaluation phase. We modified the protocol in [21] for the case of different moduli as Boudot and Traoré [4] did to obtain efficient publicly verifiable secret sharing schemes.

To obtain robustness, we first modify the dealer phase of the Asmuth-Bloom SSS and add the constraint that

$$p_i = 2m_i + 1$$

be a prime for each  $1 \leq i \leq n$ . These values will be the moduli used to construct/verify the *proof of correctness* for each user. The robustness extension described below can be used to make the CRT-based threshold RSA signature scheme in Section 2 robust. We only give the additions for the robustness extension here since the other phases are the same.

- *Setup*: Use Asmuth-Bloom SSS for sharing  $d$  with  $m_0 = \phi(N)$ . Let  $g_i$  be an element of order  $m_i$  in  $\mathbb{Z}_{p_i}^*$ . Broadcast  $g_i$  and the public verification data

$$v_i = g_i^{y_i} \bmod p_i$$

for each user  $i, 1 \leq i \leq n$ .

- *Generating the proof of correctness:* Let  $w$  be the hashed message to be signed and suppose the range of the hash function is  $\mathbb{Z}_N^*$ . Assume a coalition  $\mathcal{S}$  of size  $t$  participated in the signing phase. Let  $h : \{0, 1\}^* \rightarrow \{0, \dots, 2^{L_1} - 1\}$  be a hash function where  $L_1$  is another security parameter. Let

$$w' = w^{M_{S \setminus \{i\}}} \bmod N,$$

$$v'_i = v_i^{M'_{S,i}} \bmod p_i,$$

$$z_i = y_i M'_{S,i} \bmod m_i.$$

Each user  $i \in \mathcal{S}$  first computes

$$W = w'^r \bmod N,$$

$$G = g_i^r \bmod p_i$$

where  $r \in_R \{0, \dots, 2^{L(m_i)+2L_1}\}$ . Then he computes the proof as

$$\sigma_i = h(w', g_i, s_i, v'_i, W, G),$$

$$D_i = r + \sigma_i z_i \in \mathbb{Z}$$

and sends the proof  $(\sigma_i, D_i)$  along with the partial signature  $s_i$ .

- *Verifying the proof of correctness:* The proof  $(\sigma_i, D_i)$  for the  $i$ th user can be verified by checking

$$\sigma_i \stackrel{?}{=} h(w', g_i, s_i, v'_i, w'^{D_i} s_i^{-\sigma_i} \bmod N, g_i^{D_i} v_i'^{-\sigma_i} \bmod p_i). \quad (7)$$

Note that the above scheme can also be used to obtain a robust threshold RSA decryption scheme. Since RSA signature and decryption functions are mostly identical, we omit the details.

### 3.1 Security Analysis

Here we will prove that the proposed threshold RSA signature scheme is secure (i.e. existentially non-forgable against an adaptive chosen message attack), provided that the RSA problem is intractable (i.e. RSA function is a one-way trapdoor function [7]). 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 cannot corrupt another user in the course of an attack, i.e., the adversary is static in that sense.

First we will analyze the proof of correctness. For generating and verifying the proof of correctness, the following properties holds:

- *Completeness*: If the  $i$ th user is honest then the proof succeeds since

$$\begin{aligned} w'^{D_i} s_i^{-\sigma_i} &= w'^r \pmod{N}, \\ g_i^{D_i} v_i'^{-\sigma_i} &= g_i^r \pmod{p_i}. \end{aligned}$$

- *Soundness*: To prove the soundness, we will use a lemma by Poupard and Stern [17] which states that if the prover knows  $(a, b, \sigma, \sigma', D, D')$  such that  $a^D b^\sigma \equiv a^{D'} b^{\sigma'} \pmod{K}$  for an integer  $K$ , then he knows the discrete logarithm of  $b$  in base  $a$  unless he knows the factorization of  $K$ .

Let us define  $\Psi : \mathbb{Z}_{Np_i}^* \rightarrow \mathbb{Z}_N^* \times \mathbb{Z}_{p_i}^*$  be the CRT isomorphism, i.e.,  $x \rightarrow (x \pmod{N}, x \pmod{p_i})$  for  $x \in \mathbb{Z}_{Np_i}^*$ . Note that  $\gcd(N, p_i) = 1$ . Let  $g = \Psi^{-1}(w', g_i)$ ,  $v = \Psi^{-1}(s_i, v_i')$  and  $\tau = \Psi^{-1}(W, G)$ . Given  $W$  and  $G$ , if the  $i$ th user can compute valid proofs  $(\sigma, D)$  and  $(\sigma', D')$  then we have

$$\tau = g^D v^\sigma \pmod{Np_i} = g^{D'} v^{\sigma'} \pmod{Np_i}$$

and according to the lemma above, the  $i$ th user knows  $u_i$  unless he can completely factor  $Np_i$ . Since the factorization of  $N$  is secret we can say that if the proof is a valid proof then the discrete logarithms are equal in  $\text{mod } m_i$  and the prover knows this discrete logarithm. Hence, an adversary cannot impersonate a user without knowing his share. Similar to Boudot and Treore [4], a range check on  $D_i$  might be necessary while verifying the proof of correctness to detect incorrect partial signatures from users with valid shares.

- *Zero-Knowledge Simulatability*: To prove the zero-knowledge simulatability, we will use the random oracle model for the hash function  $h$  and construct a simple simulator. When an uncorrupted user wants to create a proof  $(\sigma_i, D_i)$  for a message  $w$  and partial signature  $s_i$ , the simulator returns

$$\sigma_i \in_R \{0, \dots, 2^{L_1} - 1\}$$

and

$$D_i \in_R \{0, \dots, 2^{L(m_i)+2L_1} - 1\}$$

and sets the value of the oracle at

$$(w', g_i, s_i, v_i', w'^{D_i} s_i^{-\sigma_i} \pmod{N}, g_i^{D_i} v_i'^{-\sigma_i} \pmod{p_i})$$

as  $\sigma_i$ . Note that, the value of the random oracle is not defined at this point but with negligible probability. When a corrupted user queries the oracle, if the value of the oracle was already set the simulator returns that value otherwise it returns a random one. It is obvious that the distribution of the output of the simulator is statistically indistinguishable from the real output.

To reduce the security of the proposed threshold RSA signature scheme to the security of the standard RSA signature scheme, the following proof constructs another simulator.

**Theorem 1.** *Given that the standard RSA signature scheme is secure, the threshold RSA signature scheme is robust and secure under the static adversary model.*

*Proof.* To reduce the problem of breaking the standard RSA signature scheme to breaking the proposed threshold scheme, we will simulate the threshold protocol with no information on the secret where the output of the simulator is indistinguishable from the adversary's point of view. Afterwards, we will show that the secrecy of the private key  $d$  is not disrupted by the values obtained by the adversary. Thus, if the threshold RSA scheme is not secure, i.e., an adversary who controls  $t - 1$  users can forge signatures in the threshold scheme, one can use this simulator to forge a signature in the standard RSA scheme.

Let  $\mathcal{S}'$  denote the set of users controlled by the adversary. To simulate the adversary's view, the simulator first selects a random interval  $I = [a, b)$  from  $\mathbb{Z}_M$ ,  $M = \prod_{i=1}^t m_i$ . The start point  $a$  is randomly chosen from  $\mathbb{Z}_M$  and the end point is computed as  $b = a + m_0 M_{\mathcal{S}'}$ . Then, the shares of the corrupted users are computed as  $y_j = a \bmod m_j$  for  $j \in \mathcal{S}'$ . Note that, these  $t - 1$  shares are indistinguishable from random ones due to (1) and the improved perfectness condition. Although the simulator does not know the real value of  $d$ , it is guaranteed that for all possible  $d$ , there exists a  $y \in I$  which is congruent to  $y_j \pmod{m_j}$  and to  $d \pmod{m_0}$ .

Since we have a  $(t, n)$ -threshold scheme, given a valid RSA signature  $(s, w)$ , the partial signature  $s_i$  for a user  $i \notin \mathcal{S}'$  can be obtained by

$$s_i = s \kappa^{-\delta_{\mathcal{S}}} \prod_{j \in \mathcal{S}'} (w^{u_j})^{-1} \bmod N$$

where  $\mathcal{S} = \mathcal{S}' \cup \{i\}$ ,  $\kappa = w^{-M_{\mathcal{S}}} \bmod N$  and  $\delta_{\mathcal{S}}$  is equal to either  $\lfloor \frac{\sum_{j \in \mathcal{S}'} u_j}{M_{\mathcal{S}}} \rfloor + 1$  or  $\lfloor \frac{\sum_{j \in \mathcal{S}'} u_j}{M_{\mathcal{S}}} \rfloor$ . The value of  $\delta_{\mathcal{S}}$  is important because it carries information on  $y$ . Let  $U = \sum_{j \in \mathcal{S}'} u_j$  and  $U_{\mathcal{S}} = U \bmod M_{\mathcal{S}}$ . One can find whether  $y$  is greater than  $U_{\mathcal{S}}$  or not by looking at  $\delta_{\mathcal{S}}$ :

$$\begin{aligned} y < U_{\mathcal{S}} & \text{ if } \delta_{\mathcal{S}} = \lfloor U/M_{\mathcal{S}} \rfloor + 1, \\ y \geq U_{\mathcal{S}} & \text{ if } \delta_{\mathcal{S}} = \lfloor U/M_{\mathcal{S}} \rfloor. \end{aligned}$$

Since the simulator does not know the real value of  $y$ , to determine the value of  $\delta_{\mathcal{S}}$ , the simulator acts according to the interval randomly chosen at the beginning of the simulation.

$$\delta_{\mathcal{S}} = \begin{cases} \lfloor U/M_{\mathcal{S}} \rfloor + 1, & \text{if } a < U_{\mathcal{S}} \\ \lfloor U/M_{\mathcal{S}} \rfloor, & \text{if } a \geq U_{\mathcal{S}} \end{cases} \quad (8)$$

It is obvious that, the value of  $\delta_{\mathcal{S}}$  is indistinguishable from the real case if  $U_{\mathcal{S}} \notin I$ . Now, we will prove that the  $\delta_{\mathcal{S}}$  values computed by the simulator does not disrupt the indistinguishability from the adversary's point of view. First of all, there are  $(n - t + 1)$  possible  $\delta_{\mathcal{S}}$  computed by using  $U_{\mathcal{S}}$  since all the operations

in the exponent depend on the coalition  $\mathcal{S}$  alone. If none of the  $U_{\mathcal{S}}$  values lies in  $I$ , the  $\delta_{\mathcal{S}}$  values observed by the adversary will be indistinguishable from a real execution of the protocol. Using this observation, we can prove that no information about the private key is obtained by the adversary.

Observing the  $t - 1$  randomly generated shares, there are  $m_0 = \phi(N)$  candidates in  $I$  for  $y$  which satisfy  $y_j = y \bmod m_j$  for all  $j \in \mathcal{S}'$ . These  $m_0$  candidates have all different remainders modulo  $m_0$  since  $\gcd(M_{\mathcal{S}'}, m_0) = 1$ . So, exactly one of the remainders is equal to the private key  $d$ . If  $U_{\mathcal{S}} \notin I$  for all  $\mathcal{S}$ , given an  $s_i$ , the shared value  $y$  can be equal to any of these  $m_0$  candidates hence any two different values of the secret key  $d$  will be indistinguishable from adversary's point of view. In our case, this happens with all but negligible probability. First, observe that  $U_{\mathcal{S}} \equiv 0 \bmod m_i$  and there are  $m_0 M_{\mathcal{S}'}/m_i$  multiples of  $m_i$  in  $I$ . Thus, the probability of  $U_{\mathcal{S}} \notin I$  for a coalition  $\mathcal{S}$  is equal to  $\left(1 - \frac{m_0 M_{\mathcal{S}'}/m_i}{M_{\mathcal{S}'}}\right) = \left(1 - \frac{m_0 M_{\mathcal{S}'}}{M_{\mathcal{S}}}\right)$ . According to (1),  $m_i > m_0^2$  for all  $i$  hence the probability of  $U_{\mathcal{S}} \notin I$  for all possible  $\mathcal{S}$  is less than  $\left(1 - \frac{1}{m_0}\right)^{n-t+1}$ , which is almost surely 1 for  $m_0 \gg n$ .

The simulator computes the public verification data of the users in  $\mathcal{S}'$  as  $v_j = g^{y_j} \bmod p_j$  for  $j \in \mathcal{S}'$ . For other users  $i \notin \mathcal{S}'$ , the simulator chooses a random integer  $y_i \in_R \mathbb{Z}_{m_i}$  and sets  $v_i = g^{y_i} \bmod p_i$ . Note that  $\gcd(N, p_i) = 1$ . So the public verification data generated by the simulator are computationally indistinguishable from the real ones.

Consequently, the output of the simulator is indistinguishable from a real instance from the adversary's point of view, and hence the simulator can be used to forge a signature in the standard RSA scheme if the threshold RSA scheme can be broken.  $\square$

## 4 Robustness in Other CRT-Based Threshold Schemes

The robustness extension given in Section 3 can be applied to other CRT-based threshold schemes as well. Here we describe how to adapt the extension to the CRT-based threshold Paillier and ElGamal function sharing schemes.

### 4.1 Robust Sharing of the Paillier Decryption Function

Paillier's probabilistic cryptosystem [16] is a member of a different class of cryptosystems where the message is used in the exponent of the encryption operation. The description of the cryptosystem is as follows:

- *Setup*: Let  $N = pq$  be the product of two large primes and  $\lambda = \text{lcm}(p-1, q-1)$ . Choose a random  $g \in \mathbb{Z}_{N^2}$  such that the order of  $g$  is a multiple of  $N$ . The public and private keys are  $(N, g)$  and  $\lambda$ , respectively.
- *Encryption*: Given a message  $w \in \mathbb{Z}_N$ , the ciphertext  $c$  is computed as

$$c = g^{w+rN} \bmod N^2$$

where  $r$  is a random number from  $\mathbb{Z}_N$ .

- *Decryption*: Given a ciphertext  $c \in \mathbb{Z}_{N^2}$ , the message  $w$  is computed as

$$w = \frac{L(c^\lambda \bmod N^2)}{L(g^\lambda \bmod N^2)} \bmod N$$

where  $L(x) = \frac{x-1}{N}$ , for  $x \equiv 1 \pmod{N}$ .

By using the combine-and-correct approach, Kaya and Selçuk proposed a threshold version of the Paillier's cryptosystem [14]. As in threshold RSA, the decryption coalition needs to compute an exponentiation,  $s = c^\lambda \bmod N^2$ , where the exponent  $\lambda$  is shared by Asmuth-Bloom SSS in the setup phase. Hence, similar to RSA, the partial result  $s_i$  of the  $i$ th user is equal to  $s_i = c^{u_i} \bmod N^2$ . The robustness extension can be applied to the Paillier cryptosystem as follows:

- *Setup*: Use Asmuth-Bloom SSS for sharing  $\lambda$  with  $m_0 = \phi(N^2) = N\phi(N)$ . Let  $g_i \in \mathbb{Z}_{p_i}^*$  be an element with order  $m_i$  in  $\mathbb{Z}_{p_i}^*$ . Broadcast the public verification data  $g_i$  and

$$v_i = g_i^{y_i} \bmod p_i$$

for each user  $i$ ,  $1 \leq i \leq n$ .

- *Generating the proof of correctness*: Let  $h : \{0, 1\}^* \rightarrow \{0, \dots, 2^{L_1} - 1\}$  be a hash function where  $L_1$  is another security parameter. Let

$$c' = c^{M_{S \setminus \{i\}}} \bmod N^2,$$

$$v'_i = v_i M'_{S,i} \bmod p_i,$$

$$z_i = y_i M'_{S,i} \bmod m_i.$$

Each user  $i \in S$  first computes

$$W = c'^r \bmod N^2,$$

$$G = g_i^r \bmod p_i$$

where  $r \in_R \{0, \dots, 2^{L(m_i)+2L_1}\}$ . Then he computes the proof as

$$\sigma_i = h(c', g_i, s_i, v'_i, W, G),$$

$$D_i = r + \sigma_i z_i \in \mathbb{Z}$$

and sends the proof  $(\sigma_i, D_i)$  along with the partial decryption  $s_i$ .

- *Verifying the proof of correctness*: The proof  $(\sigma_i, D_i)$  for the  $i$ th user can be verified by checking

$$\sigma_i \stackrel{?}{=} h(c', g_i, s_i, v'_i, c'^{D_i} s_i^{-\sigma_i} \bmod N, g_i^{D_i} v'_i^{-\sigma_i} \bmod p_i). \quad (9)$$

If the  $i$ th user is honest then the proof succeeds since  $c'^{D_i} s_i^{-\sigma_i} = c'^r \bmod N^2$  and  $g_i^{D_i} v'_i^{-\sigma_i} = g_i^r \bmod p_i$ . The soundness property can be proved with a proof similar to the proof of Theorem 1. Note that  $\gcd(N^2, p_i) = 1$  for all users and  $\phi(N^2) = N\phi(N)$  is secret. A similar proof can be given for the zero knowledge simulatability as the one in Section 3.1.

## 4.2 Robust Sharing of the ElGamal Decryption Function

The ElGamal cryptosystem [13] is another popular public key scheme with the following description:

- *Setup*: Let  $p$  be a large prime and  $g$  be a generator of  $\mathbb{Z}_p^*$ . Choose a random  $\alpha \in \{1, \dots, p-1\}$  and compute  $\beta = g^\alpha \bmod p$ .  $(\beta, g, p)$  and  $\alpha$  are the public and private keys, respectively.
- *Encryption*: Given a message  $w \in \mathbb{Z}_p$ , the ciphertext  $c = (c_1, c_2)$  is computed as

$$\begin{aligned} c_1 &= g^k \bmod p, \\ c_2 &= \beta^k w \bmod p \end{aligned}$$

where  $k$  is a random integer in  $\{1, \dots, p-1\}$ .

- *Decryption*: Given a ciphertext  $c$ , the message  $w$  is computed as

$$w = (c_1^\alpha)^{-1} c_2 \bmod p.$$

Adapting our robustness extension to the threshold ElGamal scheme given in [14] is slightly more complicated than it is for the Paillier's cryptosystem, because  $\phi(p) = p-1$  is public. A simple solution for this problem is to extend the modulus to  $N = pq$  where  $p = 2p' + 1$  and  $q = 2q' + 1$  are safe primes. There exist versions of the ElGamal encryption scheme in the literature with a composite modulus instead of  $p$ . For example, Wei et al. [22] modified the standard ElGamal scheme to obtain a hidden-order ElGamal scheme. They proved that their scheme is as secure as each of the standard RSA and ElGamal cryptosystems. Here we give the description of a robust, CRT-based threshold scheme for Wei et al.'s version of the ElGamal encryption.

- *Setup*: In the ElGamal setup phase, choose  $p = 2p' + 1$  and  $q = 2q' + 1$  be large primes such that  $p'$  and  $q'$  are also prime numbers. Let  $N = pq$  and let  $g_p$  and  $g_q$  be generators of  $\mathbb{Z}_p^*$  and  $\mathbb{Z}_q^*$ , respectively. Choose  $\alpha_p \in_R \mathbb{Z}_p^*$  and  $\alpha_q \in_R \mathbb{Z}_q^*$  such that  $\gcd(p-1, q-1) \mid (\alpha_p - \alpha_q)$ . The secret key  $\alpha \in \mathbb{Z}_{\lambda(N)}$  is the unique solution of the congruence system

$$\begin{aligned} \alpha &\equiv \alpha_p \pmod{p-1}, \\ \alpha &\equiv \alpha_q \pmod{q-1} \end{aligned}$$

where  $\lambda(N) = 2p'q'$  is the Carmichael number of  $N$ . Similarly, the public key  $\beta \in \mathbb{Z}_N$  is the unique solution of congruence system

$$\begin{aligned} \beta &\equiv g_p^{\alpha_p} \pmod{p}, \\ \beta &\equiv g_q^{\alpha_q} \pmod{q}. \end{aligned}$$

Let  $g$  be the unique solution of the congruence system

$$\begin{aligned} g &\equiv g_p \pmod{p}, \\ g &\equiv g_q \pmod{q} \end{aligned}$$

and  $\alpha$  and  $(\beta, g, N)$  be the private and the public keys, respectively. Note that  $\beta = g^\alpha \bmod N$ . Use Asmuth-Bloom SSS for sharing the private key  $\alpha$  with  $m_0 = 2p'q'$ . Let  $g_i \in \mathbb{Z}_{p_i}^*$  be an element with order  $m_i$  in  $\mathbb{Z}_{p_i}^*$ . Broadcast the public verification data  $g_i$  and  $v_i = g_i^{y_i} \bmod p_i$  for each user  $i$ ,  $1 \leq i \leq n$ .

- *Encryption*: Given a message  $w \in \mathbb{Z}_N$ , the ciphertext  $c = (c_1, c_2)$  is computed as

$$\begin{aligned} c_1 &= g^k \bmod N, \\ c_2 &= \beta^k w \bmod N \end{aligned}$$

where  $k$  is a random integer from  $\{1, \dots, N-1\}$ .

- *Decryption*: Let  $(c_1, c_2)$  be the ciphertext to be decrypted where  $c_1 = g^k \bmod N$  for some  $k \in \{1, \dots, N-1\}$  and  $c_2 = \beta^k w \bmod N$  where  $w$  is the message. The coalition  $\mathcal{S}$  of  $t$  users wants to obtain the message  $w = sc_2 \bmod N$  for the decryptor  $s = (c_1^\alpha)^{-1} \bmod N$ .

- *Generating the partial results*: Each user  $i \in \mathcal{S}$  computes

$$u_i = y_i M'_{\mathcal{S},i} M_{\mathcal{S} \setminus \{i\}} \bmod M_{\mathcal{S}}, \quad (10)$$

$$s_i = c_1^{-u_i} \bmod N,$$

$$\beta_i = g^{u_i} \bmod N. \quad (11)$$

- *Generating the proof of correctness*: Let  $h : \{0, 1\}^* \rightarrow \{0, \dots, 2^{L_1} - 1\}$  be a hash function where  $L_1$  is another security parameter. Let

$$c'_1 = c_1^{M_{\mathcal{S} \setminus \{i\}}} \bmod N,$$

$$v'_i = v_i^{M'_{\mathcal{S},i}} \bmod p_i,$$

$$z_i = y_i M'_{\mathcal{S},i} \bmod m_i.$$

Each user  $i \in \mathcal{S}$  first computes

$$W = c'^r_1 \bmod N,$$

$$G = g_i^r \bmod p_i$$

where  $r \in_R \{0, \dots, 2^{L(m_i)+2L_1}\}$ . Then he computes the proof as

$$\sigma_i = h(c'_1, g_i, s_i, v'_i, W, G),$$

$$D_i = r + \sigma_i z_i \in \mathbb{Z}$$

and sends the proof  $(\sigma_i, D_i)$  along with  $s_i$ .

- *Verifying the proof of correctness*: The proof  $(\sigma_i, D_i)$  for the  $i$ th user can be verified by checking

$$\sigma_i \stackrel{?}{=} h(c'_1, g_i, s_i, v'_i, c'^{D_i}_1 s_i^{-\sigma_i} \bmod N, g_i^{D_i} v'^{-\sigma_i}_i \bmod p_i).$$

- *Combining the partial results:* The incomplete decryptor  $\bar{s}$  is obtained by combining the  $s_i$  values

$$\bar{s} = \prod_{i \in \mathcal{S}} s_i \pmod N.$$

- *Correction:* The  $\beta_i$  values will be used to find the exponent which will be used to correct the incomplete decryptor. Compute the incomplete public key  $\bar{\beta}$  as

$$\bar{\beta} = \prod_{i \in \mathcal{S}} \beta_i \pmod N. \tag{12}$$

Let  $\kappa_s = c_1^{M_S} \pmod N$  and  $\kappa_\beta = g^{-M_S} \pmod N$  be the *correctors* for  $s$  and  $\beta$ , respectively. The corrector exponent  $\delta$  is obtained by trying

$$\bar{\beta} \kappa_\beta^j \stackrel{?}{\equiv} \beta \pmod N \tag{13}$$

for  $0 \leq j < t$ .

- *Extracting the message:* Compute the message  $w$  as

$$\begin{aligned} s &= \bar{s} \kappa_s^\delta \pmod N, \\ w &= s c_2 \pmod N. \end{aligned}$$

where  $\delta$  denotes the value of  $j$  that satisfies (13).

As in the case of RSA, the decryptor  $\bar{s}$  is *incomplete* since we need to obtain  $y = \sum_{i \in \mathcal{S}} u_i \pmod{M_S}$  as the exponent of  $c_1^{-1}$ . Once this is achieved,  $(c_1^{-1})^y \equiv (c_1^{-1})^\alpha \pmod N$  since  $y = \alpha + 2Ap'q'$  for some  $A$ .

When the equality in (13) holds we know that  $\beta = g^\alpha \pmod N$  is the correct public key. This equality must hold for one  $j$  value, denoted by  $\delta$ , in the given interval since the  $u_i$  values in (10) and (11) are first reduced modulo  $M_S$ . So, combining  $t$  of them will give  $\alpha + am_0 + \delta M_S$  in the exponent in (12) for some  $\delta \leq t - 1$ . Thus in (12), we obtained

$$\bar{\beta} = g^{\alpha + am_0 + \delta M_S} \pmod N \equiv g^{\alpha + \delta M_S} = \beta g^{\delta M_S} = \beta \kappa_\beta^{-\delta} \pmod N$$

and for  $j = \delta$  equality must hold. Actually, in (12) and (13), our purpose is not to compute the public key since it is already known. We want to find the corrector exponent  $\delta$  in order to obtain  $s$ , which is equal to the one used to obtain  $\beta$ . This equality can be seen as follows:

$$\begin{aligned} s &\equiv c_1^{-\alpha} = \beta^{-r} \\ &= \left( g^{-(\alpha + (\delta - \delta)M_S)} \right)^r \\ &= c_1^{-(\alpha + am_0 + \delta M_S)} (c_1^{M_S})^\delta = \bar{s} \kappa_s^\delta \pmod N \end{aligned}$$

If the  $i$ th user is honest then the proof succeeds since  $c_1^{D_i} s_i^{-\sigma_i} = c_1^{r_i} \pmod N$  and  $g_i^{D_i} v_i^{-\sigma_i} = g_i^{r_i} \pmod{p_i}$ . The soundness property can be proved with a proof similar to the one in Section 3.1. Note that  $\gcd(N, p_i) = 1$  for all users and  $\lambda(N) = 2p'q'$  is secret. A similar proof can be given for the zero knowledge simulatability as the one in Section 3.1. We omit the security proof here since the structure of the simulator is very similar to the one in Theorem 1 of Section 3.1.

## 5 Conclusion

In this paper, we proposed robust threshold RSA, Paillier and ElGamal schemes based on the Asmuth-Bloom SSS. Previous solutions for robust function sharing schemes were based on the Shamir's SSSs [10,15,19,21]. To the best of our knowledge, the schemes described in this paper are the first robust and secure FSSs using a CRT-based secret sharing. The ideas presented in this paper can be used to obtain other robust FSSs based on the CRT.

## References

1. Asmuth, C., Bloom, J.: A modular approach to key safeguarding. *IEEE Trans. Information Theory* 29(2), 208–210 (1983)
2. Bellare, M., Rogaway, P.: Random oracles are practical: a paradigm for designing efficient protocols. In: *Proc. of First ACM Conference on Computer and Communications Security*, pp. 62–73 (1993)
3. Blakley, G.: Safeguarding cryptographic keys. In: *Proc. of AFIPS National Computer Conference* (1979)
4. Boudot, F., Traoré, J.: Efficient publicly verifiable secret sharing schemes with fast or delayed recovery. In: Varadharajan, V., Mu, Y. (eds.) *ICICS 1999*. LNCS, vol. 1726, pp. 87–102. Springer, Heidelberg (1999)
5. Chaum, D., Evertse, J.H., Van De Graaf, J.: An improved protocol for demonstrating possession of discrete logarithm and some generalizations. In: Price, W.L., Chaum, D. (eds.) *EUROCRYPT 1987*. LNCS, vol. 304, pp. 127–141. Springer, Heidelberg (1988)
6. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) *CRYPTO 1992*. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg (1993)
7. Cramer, R., Shoup, V.: Signature schemes based on the strong RSA assumption. *ACM Trans. Inf. Syst. Secur.* 3(3), 161–185 (2000)
8. Desmedt, Y.: Some recent research aspects of threshold cryptography. In: Okamoto, E. (ed.) *ISW 1997*. LNCS, vol. 1396, pp. 158–173. Springer, Heidelberg (1998)
9. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) *CRYPTO 1989*. LNCS, vol. 435, pp. 307–315. Springer, Heidelberg (1990)
10. Desmedt, Y., Frankel, Y.: Shared generation of authenticators and signatures. In: Feigenbaum, J. (ed.) *CRYPTO 1991*. LNCS, vol. 576, pp. 457–469. Springer, Heidelberg (1992)
11. Desmedt, Y., Frankel, Y.: Homomorphic zero-knowledge threshold schemes over any finite abelian group. *SIAM Journal on Discrete Mathematics* 7(4), 667–679 (1994)
12. Ding, C., Pei, D., Salomaa, A.: *Chinese Remainder Theorem: Applications in Computing, Coding, Cryptography*. World Scientific, Singapore (1996)
13. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Information Theory* 31(4), 469–472 (1985)
14. Kaya, K., Selçuk, A.A.: Threshold cryptography based on Asmuth-Bloom secret sharing. *Information Sciences* 177(19), 4148–4160 (2007)
15. Lysyanskaya, A., Peikert, C.: Adaptive security in the threshold setting: From cryptosystems to signature schemes. In: Boyd, C. (ed.) *ASIACRYPT 2001*. LNCS, vol. 2248, pp. 331–350. Springer, Heidelberg (2001)

16. Paillier, P.: Public key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
17. Poupard, G., Stern, J.: Security analysis of a practical on the fly authentication and signature generation. In: Nyberg, K. (ed.) EUROCRYPT 1998. LNCS, vol. 1403, pp. 422–436. Springer, Heidelberg (1998)
18. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public key cryptosystems. *Comm. ACM* 21(2), 120–126 (1978)
19. De Santis, A., Desmedt, Y., Frankel, Y., Yung, M.: How to share a function securely? In: Proc. of STOC 1994, pp. 522–533 (1994)
20. Shamir, A.: How to share a secret? *Comm. ACM* 22(11), 612–613 (1979)
21. Shoup, V.: Practical threshold signatures. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 207–220. Springer, Heidelberg (2000)
22. Wei, W., Trung, T., Magliveras, S., Hoffman, F.: Cryptographic primitives based on groups of hidden order. *Tatra Mountains Mathematical Publications* 29, 147–155 (2004)