Available online at www.sciencedirect.com
ScienceDirect
Information Sciences 177 (2007) 4148–4160
INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL
www.elsevier.com/locate/ins

Threshold cryptography based on Asmuth–Bloom secret sharing [☆,☆☆]

Kamer Kaya [*], Ali Aydın Selçuk

Department of Computer Engineering, Bilkent University, Ankara, 06800, Turkey

Received 26 October 2006; received in revised form 31 January 2007; accepted 6 April 2007

Abstract

In this paper, we investigate how threshold cryptography can be conducted with the Asmuth–Bloom secret sharing scheme and present three novel function sharing schemes for RSA, ElGamal and Paillier cryptosystems. To the best of our knowledge, these are the first provably secure threshold cryptosystems realized using the Asmuth–Bloom secret sharing. Proposed schemes are comparable in performance to earlier proposals in threshold cryptography.
© 2007 Elsevier Inc. All rights reserved.

Keywords: Threshold cryptography; Function sharing schemes; Asmuth–Bloom secret sharing; RSA; ElGamal; Paillier

1. Introduction

Threshold cryptography 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. Well-known secret sharing schemes (SSS) in the literature include Shamir [25] based on polynomial interpolation, Blakley [5] based on hyperplane geometry, and Asmuth–Bloom [2] based on the Chinese Remainder Theorem.

A further requirement of a threshold cryptosystem can be that the subject function (e.g., a digital signature) should be computable without the involved parties disclosing their secret shares. This is known as 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 the individual secrets. Several protocols for function sharing [6,8–11,16,24,26] have been proposed in the literature. Nearly all existing solutions for function sharing have been based on the Shamir SSS [25].

[☆] This work is supported in part by the Turkish Scientific and Technological Research Agency (TÜBİTAK), under grant number EEEAG-105E065.
[☆☆] A preliminary version of this paper was presented in ISCIS'06, 21st International Symposium on Computer and Information Sciences.
[*] Corresponding author. Tel.: +90 312 290 1350; fax: +90 312 266 4047.
E-mail addresses: kamer@cs.bilkent.edu.tr (K. Kaya), selcuk@cs.bilkent.edu.tr (A.A. Selçuk).

0020-0255/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.ins.2007.04.008

In this paper, we show how sharing of cryptographic functions can be securely achieved using the Asmuth–Bloom secret sharing scheme. We give three novel FSSs, one for the RSA [23], one for the ElGamal decryption [13] and the other for the Paillier decryption [21] functions. These public key cryptosystems have several interesting properties useful in various applications [1,3,14,18,19]. The proposed schemes are provably secure and to the best of our knowledge they are the first realization of function sharing based on the Asmuth–Bloom SSS.

The organization of the paper is as follows: in Section 2, we give an overview of threshold cryptography and review the existing secret and function sharing schemes in the literature. We discuss the Asmuth–Bloom SSS in detail in Section 3 and our modifications on the basic scheme in Section 4. In Sections 5–7, we describe the FSSs for RSA, ElGamal and Paillier cryptosystems respectively, and prove their security features. After analyzing the efficiency of the proposed schemes in Section 8, we conclude the paper in Section 9.

2. Background

Constructing threshold schemes for secret and function sharing is the main research area in threshold cryptography. These problems have been studied for many years and several solutions have been proposed.

2.1. Secret sharing schemes

The problem of secret sharing and the first solutions to it were introduced independently by Shamir [25] and Blakley [5] in 1979. A (t,n)-secret sharing scheme is used to distribute a secret d among n people such that any coalition of size t or more can construct d but smaller coalitions cannot. Furthermore, an SSS is said to be *perfect* if coalitions smaller than t cannot obtain *any* information on d; i.e., the candidate space for d cannot be reduced even by one candidate by using $t-1$ or fewer shares.

The first scheme for sharing a secret was proposed by Shamir [25] based on polynomial interpolation. To obtain a (t,n) secret sharing, a random polynomial $f(x) = a_{t-1}x^{t-1} + a_{t-2}x^{t-2} + \cdots + a_0$ is generated over $\mathbb{Z}_p[x]$ where p is a prime number and $a_0 = d$ is the secret. The share of the ith party is $y_i = f(i)$, $1 \leqslant i \leqslant n$. If t or more parties come together, they can construct the polynomial by Lagrange interpolation and obtain the secret, but any smaller coalitions cannot.

Another interesting SSS is the scheme proposed by Blakley [5]. In a t dimensional space, a system of t non-parallel, non-degenerate hyperplanes intersect at a single point. In Blakley's scheme, a point in the t dimensional space (or, its first coordinate) is taken as the secret and each party is given a hyperplane passing through that point. When t users come together, they can uniquely identify the secret point, but smaller coalitions cannot.

A fundamentally different SSS is the scheme of Asmuth and Bloom [2], which shares a secret among the parties using modular arithmetic and reconstructs it by the Chinese Remainder Theorem. We describe this scheme in detail in Section 3.

2.2. Function sharing schemes

Threshold function sharing problem was first introduced by Desmedt and Frankel [9] in 1989. In a (t,n) function sharing scheme, a key-dependent function is distributed among n people such that any coalition of size t or more can evaluate the function but smaller coalitions cannot. When a coalition $\mathscr{S}$ is to evaluate the function, the ith user in $\mathscr{S}$ computes his own partial result by using his share y_i and sends it to the combiner to evaluate the function. The combiner must be honest while combining the partial results but can be curious and try to find the secret shares. This is not a problem since the user shares are not disclosed to the combiner.

FSSs are typically used to distribute the private key operations in a public key cryptosystem (i.e., the decryption and signature operations) among several parties. Sharing a private key operation in a threshold fashion requires first choosing a suitable SSS to share the private key. Then the subject function must be arranged according to this SSS such that combining the partial results from any t parties will yield the operation's result correctly. This is usually a challenging task and requires some ingenious techniques.

Several solutions for sharing the RSA, ElGamal and Paillier private key operations have been proposed in the literature [8–11,14,15,17,24,26]. Almost all of these schemes are based on the Shamir SSS, with the only exception of one scheme in [9] based on Blakley. The additive nature of the Lagrange interpolation used in the combiner phase of Shamir's scheme makes it a suitable choice for function sharing, but it also provides several challenges. One of the most significant challenges is the computation of inverses in $\mathbb{Z}_{\phi(N)}$ for sharing the RSA function where $\phi(N)$ should not be known by the users. The first solution to this problem was proposed by Desmedt [8], which solved the problem by making the dealer compute all potentially needed inverses at the setup time and distribute them to users mixed with the shares. A more elegant solution was found a few years later by De Santis et al. [24]. They carried the arithmetic into a cyclotomic extension of $\mathbb{Z}$, which enabled computing the inverses without knowing $\phi(N)$. Finally, a very practical and ingenious solution was given by Shoup [26] where he removed the need of taking inverses in Lagrange interpolation altogether.

Although using Shamir's SSS for sharing the ElGamal signature and decryption functions has its own unique problems, the modular inverse computation problem is relatively easier than that in RSA since all of the operations are done in mod p where p is a public prime hence $\phi(p) = p - 1$ is also public. Practical FSSs were proposed in [9,15] for ElGamal signature and decryption functions.

Shoup's practical RSA scheme inspired similar works on different cryptosystems. Fouque et al. [14] proposed a similar threshold solution for the Paillier cryptosystem and used it in e-voting and lottery schemes. Later, Lysyanskaya and Peikert [17] improved this worked and obtained a threshold Paillier encryption scheme secure under the adaptive security model.

To the best of our knowledge, so far no secure function sharing schemes based on the Asmuth–Bloom SSS have been proposed in the literature. We show in this paper that the Asmuth–Bloom scheme in fact can also be a suitable choice for function sharing, and the fundamental challenges of the other schemes discussed above do not exist for the Asmuth–Bloom scheme.

3. Asmuth–Bloom secret sharing scheme

In the Asmuth–Bloom SSS, dealing and reconstructing the secret are done as follows:

- *Dealer phase*: To share a secret d among a group of n users, the dealer does the following:
  - (·) A set of pairwise relatively prime integers $m_0 < m_1 < m_2 < \cdots < m_n$, where $m_0 > d$ is a prime, are chosen such that

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

  - (·) 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 $0 \leqslant y < M$.
  - (·) The share of the ith user, $1 \leqslant i \leqslant n$, is

    $$y_i = y \bmod m_i.$$

- *Combiner phase*: Assume $\mathscr{S}$ is a coalition of t users to construct the secret. Let $M_{\mathscr{S}}$ denote $\prod_{i \in \mathscr{S}} m_i$.
  - (·) Given the system

    $$y \equiv y_i \bmod m_i$$

    for $i \in \mathscr{S}$, find y in $\mathbb{Z}_{M_{\mathscr{S}}}$ using the Chinese Remainder Theorem.
  - (·) Compute the secret as

    $$d = y \bmod m_0.$$

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

The Asmuth–Bloom SSS is close to perfect in the sense that $t-1$ or fewer shares does not narrow down the key space: assume a coalition $\mathscr{S}'$ of size $t-1$ has gathered and let y' be the unique solution for y in $Z_{M_{\mathscr{S}'}}$. According to (1), $M/M_{\mathscr{S}'} > m_0$, hence $y' + jM_{\mathscr{S}'}$ is smaller than M for $j < m_0$. Since $\gcd(m_0, M_{\mathscr{S}'}) = 1$, all $(y' + jM_{\mathscr{S}'}) \bmod m_0$ are distinct for $0 \leqslant j < m_0$, and there are m_0 of them. That is, d can be any integer from $\mathbb{Z}_{m_0}$. However, this scheme is not exactly perfect since when $t-1$ shares are known, the key candidates are not equally likely as described in Section 4. We refer the reader to a recent work by Quisquater et al. [22] for a detailed security analysis of Asmuth–Bloom and some other Chinese Remainder Based SSSs.

4. Function sharing based on the Asmuth–Bloom scheme

Several changes were needed on the basic Asmuth–Bloom scheme to make it more suitable for function sharing. In this section we describe these modifications.

In the original Asmuth–Bloom SSS, the authors proposed an iterative process to solve the system $y \equiv y_i \pmod{m_i}$. Instead, we use a non-iterative and direct solution as described in [12], which turns out to be more suitable for function sharing in the sense that it does not require interaction between parties and has an additive structure which is convenient for exponentiations. Suppose $\mathscr{S}$ is a coalition of t users gathered to construct the secret d.

(1) Let $M_{\mathscr{S}\setminus\{i\}}$ denote $\prod_{j\in S, j\neq i} m_j$ and $M'_{\mathscr{S},i}$ be the multiplicative inverse of $M_{\mathscr{S}\setminus\{i\}}$ in $\mathbb{Z}_{m_i}$, i.e.,

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

First, the ith user computes

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

(2) y is computed as

$$y = \sum_{i\in\mathscr{S}} u_i \bmod M_{\mathscr{S}}.$$

(3) The secret d is computed as

$$d = y \bmod m_0.$$

We note that, in the Asmuth–Bloom SSS, m_0 need not be a prime, and the scheme works correctly for a composite m_0 as long as m_0 is relatively prime to m_i, $1 \leqslant i \leqslant n$. Also note that m_0 need not be known during the secret construction process until the +3rd step above.

We also modified (1) as

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

in order to use it securely in the proposed FSSs. As discussed in Section 3, Eq. (1) guarantees that d can still be any integer from $\mathbb{Z}_{m_0}$ when $t-1$ or fewer shares are revealed. We also know that, for each value of d, there are either $\lfloor M/(M_{\mathscr{S}'} m_0) \rfloor$ or $\lfloor M/(M_{\mathscr{S}'} m_0) \rfloor + 1$ 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 equals these integers may not be equal. (E.g., for $M/M_{\mathscr{S}'} \approx 3m_0/2$, half of the integers in $\mathbb{Z}_{m_0}$ are twice more likely than the other half.) Eq. (2) solves this problem by guaranteeing that $M/(M_{\mathscr{S}'} m_0) > m_0$. Given that $m_0 \gg 1$, all d values are approximately equally likely.

In the FSSs described in this paper, m_i, $1 \leqslant i \leqslant n$, are known by all users, but m_0 is kept secret by the dealer.

5. Sharing of the RSA function

RSA [23] is the first and most commonly used public key cryptosystem today. Here we show how the RSA signature and decryption functions can be shared by using the Asmuth–Bloom SSS. Below, we limit our discussion to the RSA signature function since these two functions are identical and the same technique can be applied for sharing the decryption function as well. The description of the RSA signature scheme is as follows:

- *Setup*: Let $N = pq$ be the product of two large prime numbers. Choose a random $e \in \mathbb{Z}^*_{\phi(N)}$ and find its inverse d, i.e., $ed \equiv 1(\bmod\ \phi(N))$. The public and private keys are (N, e) and d, respectively.
- *Signing*: Given a hashed message $w \in \mathbb{Z}_N$, the signature s is computed as

  $$s = w^d \bmod N.$$

- *Verification*: Given a signature $s \in \mathbb{Z}_N$, the verification is done by checking

  $$w \overset{?}{=} s^e \bmod N.$$

Threshold RSA signature scheme: The following is a procedure that shares the RSA signature function among n users with the Asmuth–Bloom SSS such that when t users come together they can compute the signature:

- *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 $ed \equiv 1(\bmod \phi(N))$. Use Asmuth–Bloom SSS for sharing d with $m_0 = \phi(N) = 4p'q'$.
- *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 $\mathscr{S}$ of size t wants to obtain the signature $s = w^d \bmod N$.
  - $(\cdot)$ *Generating partial results*: Each user $i \in \mathscr{S}$ computes

    $$u_i = y_i M'_{\mathscr{S},i} M_{\mathscr{S}\setminus\{i\}} \bmod M_{\mathscr{S}},$$
    $$s_i = w^{u_i} \bmod N.$$

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

    $$\bar{s} = \prod_{i \in \mathscr{S}} s_i \bmod N. \tag{3}$$

  - $(\cdot)$ *Correction*: Let $\kappa = w^{-M_{\mathscr{S}}} \bmod N$ be the *corrector*. The incomplete signature can be corrected by trying

    $$(\bar{s}\kappa^j)^e = \bar{s}^e(\kappa^e)^j \overset{?}{\equiv} w \pmod N \tag{4}$$

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

    $$s = \bar{s}\kappa^\delta \bmod N,$$

    where δ denotes the value of j that satisfies (4).
- *Verification* is the same as the standard RSA verification.

We call the signature $\bar{s}$ generated in (3) *incomplete* since we need to obtain $y = \sum_{i\in\mathscr{S}} u_i \bmod M_{\mathscr{S}}$ 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 (4) must hold for some $j \leqslant t - 1$ since the u_i values were already reduced modulo $M_{\mathscr{S}}$. So, combining t of them in (3) will give $d + am_0 + \delta M_{\mathscr{S}}$ in the exponent for some $\delta \leqslant t - 1$. Thus in (3), we obtained

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

and for $j = \delta$, Eq. (4) 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 (4).

5.1. Security analysis

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

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

Proof 1. 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 $\mathscr{S}'$ 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 $b=a+m_0M_{\mathscr{S}'}$. Then, the shares of the corrupted users are computed as $y_j=a \bmod m_j$ for $j\in\mathscr{S}'$. Note that, these $t-1$ shares are indistinguishable from random ones due to (2) and the improved perfectness condition. Although the simulator does not know the real value of d, it is guaranteed that there exists a $y\in I$ which is congruent to $y_j(\bmod\, m_j)$ and $d(\bmod\, m_0)$ for all possible d values.

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

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

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

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

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

$$\delta_{\mathscr{S}}=\begin{cases}\lfloor U/M_{\mathscr{S}}\rfloor+1 & \text{if} \quad a<U_{\mathscr{S}},\\ \lfloor U/M_{\mathscr{S}}\rfloor & \text{if} \quad a\geqslant U_{\mathscr{S}}.\end{cases} \tag{5}$$

It is obvious that, the value of $\delta_{\mathscr{S}}$ is indistinguishable from the real case if $U_{\mathscr{S}}\notin I$. Now, we will prove that the $\delta_{\mathscr{S}}$ 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 $\delta_{\mathscr{S}}$ computed by using $U_{\mathscr{S}}$ since all the operations in the exponent depend on the coalition $\mathscr{S}$ alone. If none of the $U_{\mathscr{S}}$ values lies in I, the $\delta_{\mathscr{S}}$ 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 $j\in\mathscr{S}'$. These m_0 candidates have all different remainders modulo m_0 since $\gcd(M_{\mathscr{S}'},m_0)=1$. So, exactly one of the remainders is equal to the private key d. If $U_{\mathscr{S}}\notin I$ for all $\mathscr{S}$, 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 $U_{\mathscr{S}} \equiv 0 \bmod m_i$ and there are $m_0 M_{\mathscr{S}'}/m_i$ multiples of m_i in I. Thus, the probability of $U_{\mathscr{S}} \notin I$ for a coalition $\mathscr{S}$ is equal to $(1 - \frac{m_0 M_{\mathscr{S}'}/m_i}{M_{\mathscr{S}'}}) = (1 - \frac{m_0 M_{\mathscr{S}'}}{M_{\mathscr{S}}})$. According to (2), $m_i > m_0^2$ for all i hence the probability of $U_{\mathscr{S}} \notin I$ for all possible $\mathscr{S}$ is less than $(1 - \frac{1}{m_0})^{n-t+1}$, which is almost surely $+1$ for $m_0 \gg n$.

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. □

6. Sharing of the ElGamal decryption function

The ElGamal cryptosystem [13] is another popular public key scheme proposed by T. ElGamal in 1989. It is an inherently probabilistic and semantically secure encryption scheme. The description of the cryptosystem is as follows:

- *Setup*: Let p be a large prime and g be a generator of $\mathbb{Z}_p$. Choose a random $\alpha \in \{1, \ldots, p-1\}$ and compute $\beta = g^\alpha \bmod p$. (β, g, p) and α 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

  $$c_1 = g^r \bmod p$$
  $$c_2 = \beta^r w \bmod p$$

  where r is a random integer from $\mathbb{Z}_p$.
- *Decryption*: Given a ciphertext c, the message w is computed as

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

  ElGamal encryption scheme, like RSA, has the following multiplicative homomorphic property:

  $$E(w) \times E(w') = E(ww')$$

  for messages w and w' where E stands for the encryption function and $\times$ is the component-wise multiplication. Since the standard RSA encryption is deterministic, it is not semantically secure. One can use random padding to add semantic security as in [4]. However, this removes the homomorphic property. ElGamal does not suffer from such a problem since it is inherently semantically secure. This property makes ElGamal encryption suitable for use in threshold password authenticated key exchange protocols [1].

Threshold ElGamal encryption scheme: The following is a procedure that shares the ElGamal decryption function among n users with the Asmuth–Bloom SSS such that when t users come together they can decrypt the ciphertext:

- *Setup*: In the ElGamal setup phase, choose $p = 2q + 1$ where q is a large random prime and let $g \in \mathbb{Z}_p^*$ with order q. Choose a random $\alpha \in \{1, \ldots, p-1\}$ and compute $\beta = g^\alpha \bmod p$. Let α and (β, g, p) be the private and the public keys, respectively. Use Asmuth–Bloom SSS for sharing the private key α with $m_0 = 2q$.
- *Encryption* is the same as the standard ElGamal encryption.
- *Decryption*: Let (c_1, c_2) be the ciphertext to be decrypted where $c_1 = g^k \bmod p$ for some $k \in \{1, \ldots, p-1\}$ and $c_2 = \beta^k w$ where w is the message. The coalition $\mathscr{S}$ of t users wants to obtain the message $w = sc_2 \bmod p$ for the *decryptor* $s = (c_1^\alpha)^{-1} \bmod p$.
  - *Generating partial results*: Each user $i \in \mathscr{S}$ computes

    $$u_i = y_i M'_{\mathscr{S},i} M_{\mathscr{S}\setminus\{i\}} \bmod M_{\mathscr{S}}, \tag{6}$$
    $$s_i = c_1^{-u_i} \bmod p,$$
    $$\beta_i = g^{u_i} \bmod p. \tag{7}$$

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

  $$\bar{s} = \prod_{i \in \mathscr{S}} s_i \bmod p.$$

- *Correction*: The β_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

  $$\bar{\beta} = \prod_{i \in \mathscr{S}} \beta_i \bmod p. \tag{8}$$

  Let $\kappa_s = c_1^{M_{\mathscr{S}}} \bmod p$ and $\kappa_\beta = g^{-M_{\mathscr{S}}} \bmod p$ be the *correctors* for s and β, respectively. The corrector exponent δ can be obtained by trying

  $$\bar{\beta}\kappa_\beta^j \overset{?}{\equiv} \beta \bmod p \tag{9}$$

  for $0 \leqslant j < t$.
- *Extracting the message*: Compute the message w as

  $$s = \bar{s}\kappa_s^\delta \bmod p,$$
  $$w = sc_2 \bmod p,$$

  where δ denotes the value of j that satisfies (9).

As in the case of RSA, the decryptor $\bar{s}$ is *incomplete* since we need to obtain $y = \sum_{i \in \mathscr{S}} u_i \bmod M_{\mathscr{S}}$ as the exponent of c_1^{-1}. Once this is achieved, $(c_1^{-1})^y \equiv (c_1^{-1})^\alpha \bmod N$ since $y = \alpha + A\phi(p)$ for some A.

When the equality in (9) holds we know that $\beta = g^\alpha \bmod p$ is the correct public key. This equality must hold for one j value, denoted by δ, in the given interval because since the u_i values in (6) and (7) are first reduced modulo $M_{\mathscr{S}}$. So, combining t of them will give $\alpha + am_0 + \delta M_{\mathscr{S}}$ in the exponent in (8) for some $\delta \leqslant t - 1$. Thus in (8), we obtained

$$\bar{\beta} = g^{\alpha + am_0 + \delta M_{\mathscr{S}}} \bmod p \equiv g^{\alpha + \delta M_{\mathscr{S}}} = \beta g^{\delta M_{\mathscr{S}}} = \beta \kappa_\beta^{-\delta} \bmod p$$

and for $j = \delta$ equality must hold. Actually, in (8) and (9), our purpose is not computing the public key since it is already known. We want to find the corrector exponent δ to obtain s, which is also equal to the one we use to obtain β. The equality can be verified as seen below:

$$\begin{aligned} s &\equiv c_1^{-\alpha} = \beta^{-r} \\ &= (g^{-(\alpha + (\delta - \delta)M_{\mathscr{S}})})^r \\ &= c_1^{-(\alpha + am_0 + \delta M_{\mathscr{S}})}(c_1^{M_{\mathscr{S}}})^\delta = \bar{s}\kappa_s^\delta \bmod p. \end{aligned}$$

6.1. Security analysis

Here, we will prove that the threshold ElGamal encryption scheme is semantically secure provided that the standard ElGamal encryption scheme is semantically secure. We refer the reader to [14] for a formal definition of the threshold semantic security.

Theorem 2. *Given that the standard ElGamal encryption scheme is semantically secure, the threshold ElGamal encryption scheme is semantically secure under the static adversary model.*

Proof 2. The structure of the proof is similar to that we did for the threshold RSA signature scheme. Let $\mathscr{S}'$ 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 $b = a + m_0 M_{\mathscr{S}'}$. Then, the shares of the corrupted users are computed as $y_j = a \bmod m_j$ for $j \in \mathscr{S}'$.

Since we have a (t,n)-threshold scheme, when we determine the y_j values for $j \in \mathscr{S}'$, the shares of other users are also determined. Although they cannot be computed easily, given a valid message-ciphertext pair $(w,(c_1,c_2))$ the partial decryptor s_i and β_i for a user $i \notin \mathscr{S}'$ can be obtained by

$$s_i = (wc_2^{-1})\kappa_s^{-\delta_{\mathscr{S}}} \prod_{j\in\mathscr{S}'} c_1^{u_j} \bmod p,$$

$$\beta_i = \beta\kappa_\beta^{-\delta_{\mathscr{S}}} \prod_{j\in\mathscr{S}'} (\beta^{u_j})^{-1} \bmod p,$$

where $\mathscr{S} = \mathscr{S}' \cup \{i\}$, $\kappa_s = c_1^{M_{\mathscr{S}}} \bmod p$, $\kappa_\beta = g^{-M_{\mathscr{S}}} \bmod p$ and $\delta_{\mathscr{S}}$ is equal to either $\left\lfloor \frac{\sum_{j\in\mathscr{S}'} u_j}{M_{\mathscr{S}}} \right\rfloor + 1$ or $\left\lfloor \frac{\sum_{j\in\mathscr{S}'} u_j}{M_{\mathscr{S}}} \right\rfloor$. We use the same ideas to choose the value of $\delta_{\mathscr{S}}$ as in the previous simulator so we skip the details and the analysis for the secrecy of the private key in the proof.

Consequently, the output of the simulator is indistinguishable from the adversary's point of view, and hence we proved that the threshold ElGamal scheme must be semantically secure if the standard one is. □

7. Sharing of the Paillier decryption function

Paillier's probabilistic cryptosystem [21] 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 λ, respectively.
- *Encryption*: Given a message $w \in \mathbb{Z}_N$, the ciphertext c is computed as
  $$c = g^w r^N \bmod N^2,$$
  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 $x \equiv 1 \bmod N$.

Paillier's encryption scheme is probabilistic and has interesting homomorphic properties:

$$E(w_1)E(w_2) = E(w_1 + w_2)$$
$$E(w)^a = E(aw)$$

for messages, w, w_1, w_2 and a random integer a where E stands for the encryption function. These homomorphic properties make this encryption scheme suitable for different applications such as secure voting and lottery protocols [3,14], DSA sharing protocols [18], and private information retrieval [19].

Threshold Paillier encryption scheme: The following is a procedure that shares the Paillier decryption function among n users with the Asmuth–Bloom SSS such that when t users come together they can decrypt the ciphertext. The setup part below is inspired by [14]:

- *Setup*: In the Paillier setup phase, choose large primes $p = 2p' + 1$ and $q = 2q' + 1$ where p' and q' are also large random primes and $\gcd(N, \phi(N)) = 1$ for $N = pq$. Let $g = (1+N)^a b^N \bmod N^2$ for random a and b from $\mathbb{Z}_N^*$. Compute $\theta = a\beta\lambda \bmod N$ for a random $\beta \in \mathbb{Z}_N^*$ where $\lambda = \text{lcm}(p-1, q-1)$ is the Carmichael number for N. Let (N,g,θ) and λ be the public and private keys, respectively. Use the Asmuth–Bloom SSS to share $\beta\lambda$ with $m_0 = N\lambda$.
- *Encryption* is the same as the standard Paillier encryption.
- *Decryption*: Let $c = g^w r^N \bmod N^2$ be the ciphertext to be decrypted for some random $r \in \mathbb{Z}_N^*$ where w is the message from $\mathbb{Z}_N$. Assume a coalition $\mathscr{S}$ of size t wants to obtain the message $w = \frac{L(c^{\beta\lambda} \bmod N^2)}{\theta} \bmod N$. We call $s = c^{\beta\lambda} \bmod N^2$ as the *decryptor*.

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

  $$u_i = y_i M'_{\mathscr{S},i} M_{\mathscr{S}\setminus\{i\}} \bmod M_{\mathscr{S}},$$
  $$s_i = c^{u_i} \bmod N^2,$$
  $$\theta_i = g^{u_i} \bmod N^2.$$

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

  $$\bar{s} = \prod_{i \in \mathscr{S}} s_i \bmod N^2.$$

- *Correction*: The θ_i values will be used to find the exponent which corrects the incomplete decryptor. Compute the incomplete $\bar{\theta}$ as

  $$\bar{\theta} = \prod_{i \in \mathscr{S}} \theta_i \bmod N^2. \tag{10}$$

  Let $\kappa_s = c^{-M_{\mathscr{S}}} \bmod N^2$ and $\kappa_\theta = g^{-M_{\mathscr{S}}} \bmod N^2$ be the *correctors* for s and θ, respectively. The corrector exponent δ can be obtained by trying

  $$\theta \overset{?}{=} L(\bar{\theta}\kappa_\theta^j \bmod N^2) \tag{11}$$

  for $0 \leqslant j < t$. Note that, for wrong corrector exponents L is undefined.
- *Extracting the message*: Compute the message w as

  $$s = \bar{s}\kappa_s^\delta \bmod N^2,$$
  $$w = \frac{L(s)}{\theta} \bmod N,$$

  where δ denotes the value for j that satisfies (11).

The decryptor $\bar{s}$ is *incomplete* and to find the corrector exponent we used a similar approach. When the equality in (11) holds we know that $\theta = a\beta\lambda \bmod N^2$ is the correct value. Also, this equality must hold for one j value, denoted by δ, in the given interval. Actually, in (10) and (11), our purpose is not computing θ since it is already known. We want to find the corrector exponent δ to obtain s, which is also equal to the one we used to obtain θ.

7.1. Security analysis

Here, we will prove that the threshold Paillier encryption scheme is semantically secure provided that the standard Paillier encryption scheme is semantically secure.

Theorem 3. *Given that the standard Paillier encryption scheme is semantically secure, the threshold Paillier encryption scheme is semantically secure under the static adversary model.*

Proof 3. The structure of the proof is similar to those we did for the previous threshold schemes. Let $\mathscr{S}'$ 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 $b = a + m_0 M_{\mathscr{S}'}$. Then, the shares of the corrupted users are computed as $y_j = a \bmod m_j$ for $j \in \mathscr{S}'$.

Since we have a (t, n)-threshold scheme, when we determine the y_j values for $j \in \mathscr{S}'$, the shares of other users are also determined. Although they cannot be computed easily, given a valid message-ciphertext pair (w, c) the decryptor share s_i and θ_i for a user $i \notin \mathscr{S}'$ can be obtained by

$$s_i = (1 + w\theta N)\kappa_s^{-\delta_{\mathscr{S}}} \prod_{j \in \mathscr{S}'} (c_1^{u_j})^{-1} \bmod N^2,$$
$$\theta_i = (1 + \theta N)\kappa_\theta^{-\delta_{\mathscr{S}}} \prod_{j \in \mathscr{S}'} (\theta^{u_j})^{-1} \bmod N^2,$$

where $\mathscr{S} = \mathscr{S}' \cup \{i\}$, $\kappa_s = c^{-M_{\mathscr{S}}} \bmod N^2$, $\kappa_\theta = g^{-M_{\mathscr{S}}} \bmod N^2$ and $\delta_{\mathscr{S}}$ is equal to either $\left\lfloor \frac{\sum_{j\in\mathscr{S}'} u_j}{M_{\mathscr{S}}} \right\rfloor + 1$ or $\left\lfloor \frac{\sum_{j\in\mathscr{S}'} u_j}{M_{\mathscr{S}}} \right\rfloor$. We use the same ideas to choose the value of $\delta_{\mathscr{S}}$ as in the previous simulator so we skip the details and the analysis for the secrecy of the private key in the proof.

Consequently, the output of the simulator is indistinguishable from the adversary's point of view, and hence we proved that the threshold Paillier scheme must be semantically secure if the standard one is. $\square$

8. Efficiency analysis of the proposed schemes

Although the proposed schemes are not more efficient than Shoup's work [26], which is the fastest threshold RSA signature scheme, they are comparable in performance. In this section, we give an efficiency analysis of the proposed schemes. First, we compare the proposed threshold RSA scheme with the basic RSA scheme in [26] in terms of share size and computation cost. For the computation cost, the dominating factor is the exponentiation operations hence we are mainly interested in the exponentiations. Note that, the cost of an exponentiation is proportional to the size of the exponent.

- *Share size*: In [26], the size of a share is approximately k bits for a k-bit modulus N. In our case, because of (2) the size of a share is about $2k$ bits for the same N.
- *Computing partial signatures*: In [26], it takes an exponentiation with a $(k + \log(n!))$-bit exponent to compute a partial signature. In the proposed scheme,

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

  is a $2kt$-bit integer. To compute it efficiently we first compute $M'_{\mathscr{S},i}$ and $r = \lfloor y_i M'_{\mathscr{S},i}/m_i \rfloor$ which are $2k$-bit integers. Now u_i is equal to

  $$u_i = M_{\mathscr{S}\setminus\{i\}}(y_i M'_{\mathscr{S},i} - r m_i)$$

  and computing the partial signature $s_i = w^{u_i} \bmod N$ needs a modular exponentiation with $2kt$-bit exponent. Note that no extra storage is needed to store u_i.
- *Combining partial signatures*: In [26], combining the partial results requires t exponentiations with approximately $\log(n!)$-bit exponents, hence the cost is $t\log(n!)$. After that these t results are multiplied to obtain the signature. In the proposed scheme, after obtaining the incomplete signature, an exponentiation with a $2kt$-bit exponent is needed to compute the corrector. Note that while computing the partial signature the ith player computes $w^{M_{\mathscr{S}\setminus\{i\}}} \bmod N$ as an intermediate value. The combiner can compute its inverse and raise it to the m_ith power to compute the corrector which requires an exponentiation with $2k$-bit exponent rather than $2kt$. After that, at most $2t$ more multiplications are required for computing the incomplete signature and checking Eq. (4).

Table 1 compares the performance of the proposed scheme with that of [26]. Although not more efficient, the proposed RSA signature scheme is comparable in performance to Shoup's scheme given that t is a small integer, which is the case in a typical application. Regarding the proposed threshold ElGamal and Pallier schemes, their complexity differs from the threshold RSA scheme only by a constant factor and hence is similar to that in Table 1.

Table 1
Comparison of the proposed threshold RSA signature scheme with Shoup's scheme [26] in terms of the share sizes, and the cost of computing and combining the partial signatures measured in terms of the total size of exponents

Criteria	Shoup's scheme	Proposed scheme
Share sizes	k	$2k$
Cost of computing partial signatures	$k + \log(n!)$	$2kt$
Cost of combining partial signatures	$t\log(n!)$	$2k$

9. Conclusion

In this paper, sharing of the RSA signature and the ElGamal and Paillier decryption functions with the Asmuth–Bloom SSS is investigated. Previous solutions for sharing these functions were traditionally based on the Shamir's and Blakley's SSSs [6,8–10,14,16,17,24,26]. To the best of our knowledge, the schemes described in this paper are the first secure FSSs that use the Asmuth–Bloom SSS.

As a future work, ways of improving the efficiency of the proposed schemes can be investigated. Especially, finding a way to compute the signatures/messages without the correction phase would be a significant improvement. Also, one can investigate how to integrate additional features like robustness [15] and proactivity [20] into the proposed schemes. The ideas presented in this paper can also be used to obtain further FSSs for different public key cryptosystems.

Acknowledgments

We thank İsmail Güloğlu for informative discussions and his comments on this paper, to Zahir Tezcan for his comments on the ElGamal threshold scheme, and to Baha Güçlü Dündar and Said Kalkan for their comments on the Paillier threshold scheme. We also thank anonymous *Information Sciences* referees for their valuable comments which significantly helped to improved the paper.

References

[1] M. Abdalla, O. Chevassut, P.-A. Fouque, D. Pointcheval, A simple threshold authenticated key exchange from short secrets, in: Proc. of ASIACRYPT 2005, LNCS, vol. 3778, Springer-Verlag, 2005, pp. 566–584.
[2] C. Asmuth, J. Bloom, A modular approach to key safeguarding, IEEE Trans. Informat. Theory 29 (2) (1983) 208–210.
[3] O. Baudron, P.-A. Fouque, D. Pointcheval, G. Poupard, J. Stern, Practical multi-candidate election system, in: Proc. of PODC 2001, 20th ACM Symposium on Principles of Distributed Computing, 2001, pp. 274–283.
[4] M. Bellare, P. Rogaway, Optimal asymmetric encryption, in: Proc. of EUROCRYPT 1994, LNCS, vol. 950, Springer-Verlag, 1994, pp. 92–111.
[5] G. Blakley, Safeguarding cryptographic keys, in: Proc. of AFIPS National Computer Conference, 1979.
[6] C.K. Chu, W.G. Tzeng, Optimal resilient threshold signatures, Informat. Sci. 177 (8) (2007) 1834–1851.
[7] R. Cramer, V. Shoup, Signature schemes based on the strong RSA assumption, ACM Trans. Informat. Syst. Security 3 (3) (2000) 161–185.
[8] Y. Desmedt, Some recent research aspects of threshold cryptography, in: Proc. of ISW '97, 1st International Information Security Workshop, LNCS, vol. 1196, Springer-Verlag, 1997, pp. 158–173.
[9] Y. Desmedt, Y. Frankel, Threshold cryptosystems, in: Proc. of CRYPTO'89, LNCS, vol. 435, Springer-Verlag, 1990, pp. 307–315.
[10] Y. Desmedt, Y. Frankel, Shared generation of authenticators and signatures, in: Proc. of CRYPTO'91, LNCS, vol. 576, Springer-Verlag, 1992, pp. 457–469.
[11] Y. Desmedt, Y. Frankel, Homomorphic zero-knowledge threshold schemes over any finite abelian group, SIAM J. Discrete Math. 7 (4) (1994) 667–679.
[12] C. Ding, D. Pei, A. Salomaa, Chinese Remainder Theorem: Applications in Computing, Coding, Cryptography, World Scientific, 1996.
[13] T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE Trans. Informat. Theory 31 (4) (1985) 469–472.
[14] P.A. Fouque, G. Poupard, J. Stern, Sharing decryption in the context of voting or lotteries, in: Proc. of FC 2000, 4th International Conference on Financial Cryptography, LNCS, vol. 1962, Springer-Verlag, 2001, pp. 90–104.
[15] R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, Robust threshold DSS signatures, Informat. Comput. 164 (1) (2001) 54–84.
[16] H.F. Huang, C.C. Chang, A novel efficient (t, n) threshold proxy signature scheme, Informat. Sci. 176 (10) (2006) 1338–1349.
[17] A. Lysyanskaya, C. Peikert, Adaptive security in the threshold setting: from cryptosystems to signature schemes, in: Proc. of ASIACRYPT 2001, LNCS, vol. 2248, Springer-Verlag, 2001, pp. 331–350.
[18] P. MacKenzie, M.K. Reiter, Two-party generation of DSA signatures, Int. J. Informat. Security 2 (3) (2004) 218–239.
[19] R. Ostrovsky, W. Skeith, Private searching on streaming data, in: Proc. of CRYPTO'05, LNCS, vol. 3621, Springer-Verlag, 2005, pp. 223–240.
[20] R. Ostrovsky, M. Yung, How to withstand mobile virus attacks, in: Proc. of 10th ACM Symposium on the Principles of Distributed Computing, ACM, 1991, pp. 51–61.
[21] P. Paillier, Public key cryptosystems based on composite degree residuosity classes, in: Proc. of EUROCRYPT 1999, LNCS, vol. 1592, Springer-Verlag, 1999, pp. 223–238.
[22] M. Quisquater, B. Preneel, J. Vandewalle, On the security of the secret sharing scheme based on the chinese remainder theorem, in: Proc. of PKC 2002, LNCS, vol. 2274, Springer-Verlag, 2002, pp. 199–210.

[23] R. Rivest, A. Shamir, L. Adleman, A method for obtaining digital signatures and public key cryptosystems, Commun. ACM 21 (2) (1978) 120–126.
[24] A. De Santis, Y. Desmedt, Y. Frankel, M. Yung, How to share a function securely? in: Proc. of STOC94, 1994, pp. 522–533.
[25] A. Shamir, How to share a secret? Commun. ACM 22 (11) (1979) 612–613.
[26] V. Shoup, Practical threshold signatures, in: Proc. of EUROCRYPT 2000, LNCS, vol. 1807, Springer-Verlag, 2000, pp. 207–220.