Threshold Paillier and Naccache-Stern Cryptosystems Based on Asmuth-Bloom Secret Sharing

Kamer Kaya[1], Baha Güçlü Dündar[2], Said Kalkan[1], and Ali Aydın Selçuk[1]

[1] Department of Computer Engineering
Bilkent University
Ankara, 06800, Turkey
[2] Institute of Applied Mathematics
Middle East Technical University
Ankara, 06531, Turkey
{kamer,skalkan,selcuk}@cs.bilkent.edu.tr, e114491@metu.edu.tr

Abstract. In threshold cryptography, function sharing schemes allow us to distribute the computation of signature and decryption functions among several parties. Recently, function sharing schemes using Asmuth-Bloom secret sharing were proposed for the RSA signature and ElGamal decryption functions by Kaya et al. In this paper, we extend their ideas and present two novel function sharing schemes for the Paillier and Naccache-Stern knapsack cryptosystems. These cryptosystems have some interesting homomorphic properties which make them useful in various protocols and applications.

1 Introduction

In public key cryptosystems, secure storage of the private key is an important problem. A key that is saved only by one person can easily be lost. Giving a copy to several people is not a good idea either since this would increase the chance of compromise significantly. To alleviate this problem, the private key can be shared among several people so that evaluation of the private key functions will require cooperation among the share holders.

The problem of sharing a sensitive secret among a group of n users so that only a sufficient number t of them can together reconstruct the secret is known as the secret sharing problem. Well-known secret sharing schemes (SSS) in the literature include Shamir [12] based on polynomial interpolation, Blakley [3] based on hyperplane geometry, and Asmuth-Bloom [2] based on the Chinese Remainder Theorem.

Secret sharing schemes have the limitation that users must reveal their shares at the time of secret reconstruction. It would be better instead to share the function computation among the users so that the result can be obtained from the partial results computed by each user without requiring them to disclose their secrets nor requiring the reconstruction of the private key. This problem is

known as the function sharing problem. Several FSSs have been proposed in the literature for different cryptosystems, generally using the Shamir's SSS [6, 5, 7, 11, 13].

Recently Kaya et al. [17] showed how function sharing can be achieved with the Asmuth-Bloom SSS and gave novel FSSs for the RSA and ElGamal cryptosystems. In this paper, we extend their ideas to the Paillier and Naccache-Stern knapsack cryptosystems and present two FSSs for the decryption operations of these public key systems.

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. In Section 3 we discuss the Asmuth-Bloom SSS in detail. In Section 4 and 5, we describe the proposed FSSs. The paper is concluded with the discussion of proposed schemes in Section 6.

2 Background

In this section, we give an overview of the field of threshold cryptography and discuss briefly some of the main secret and function sharing schemes in the literature.

2.1 Secret Sharing Schemes

The problem of secret sharing and its first solutions were introduced independently by Shamir [12] and Blakley [3] 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, a 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 [12] based on polynomial interpolation. To obtain a (t, n) secret sharing, a random polynomial $f(x) = a_{t-1}x^{t-1} + a_{t-2}x^{t-2} + \ldots + a_0$ is generated over $\mathbb{Z}_p[x]$ where $p > d, n$ is a prime and $a_0 = d$ is the secret. The share of the ith party is $y_i = f(i)$, $1 \leq i \leq n$. If t or more parties come together, they can construct the polynomial by Lagrangian interpolation and obtain the secret, but any smaller coalitions cannot.

Another interesting SSS is the scheme proposed by Blakley [3]. 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 any smaller coalition 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

Function sharing schemes were first introduced by Desmedt et al. [6] in 1989. 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 $\mathcal{S}$ is to evaluate the function, the ith user in $\mathcal{S}$ computes his own partial result by using his share y_i and sends it to a platform which combines these partial results. Unlike in a secret sharing scheme, the platform here need not be trusted since the user shares are not disclosed to the platform.

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 [6, 5, 7, 9, 11, 13, 8]. Almost all of these schemes are based on the Shamir SSS, with the only exception of one scheme in [6] based on Blakley. Lagrangian interpolation used in the secret reconstruction 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, albeit a relatively less efficient one, was proposed by Desmedt and Frankel [5], 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. [11]. 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 [13] where he removed the need of taking inverses in Lagrangian interpolation altogether by a slight modification in the RSA signature function.

The first function sharing schemes based on the Asmuth-Bloom SSS have been proposed by Kaya et al. [17], where they used the Asmuth-Bloom SSS to share the RSA and ElGamal private key operations. We show in this paper that Asmuth-Bloom SSS can also be used for sharing the Paillier and Naccache-Stern knapsack decryption functions.

3 Asmuth-Bloom Secret Sharing Scheme

The Asmuth-Bloom SSS has the secret sharing and reconstruction procedures as follows:

Secret sharing: To share a secret d among a group of n users, the dealer does the following:

1. A set of pairwise relatively prime integers $m_0 < m_1 < m_2 < \ldots < m_n$, where $m_0 > d$ is a prime, are chosen such that

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

2. Let $M = \prod_{i=1}^{t} m_i$. The dealer computes

$$y = d + am_0$$

where a is a positive integer generated randomly subject to the condition that $0 \leq y < M$.

3. The share of the ith user, $1 \leq i \leq n$, is

$$y_i = y \bmod m_i.$$

Secret construction: Assume $\mathcal{S}$ is a coalition of t users to construct the secret. Let $M_{\mathcal{S}} = \prod_{i \in \mathcal{S}} m_i$.

1. Given the system

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

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

2. Compute the secret as

$$d = y \bmod m_0.$$

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 $\mathbb{Z}_M$.

The Asmuth-Bloom SSS is a perfect sharing scheme: Assume a coalition $\mathcal{S}'$ of $t-1$ malicious users has gathered. Let y' be the unique solution for y in $Z_{M_{\mathcal{S}'}}$. According to (1), $M/M_{\mathcal{S}'} > m_0$, hence $y' + jM_{\mathcal{S}'}$ is smaller than M for $j < m_0$. Since $\gcd(m_0, M_{\mathcal{S}'}) = 1$, all $(y' + jM_{\mathcal{S}'}) \bmod m_0$ are distinct for $0 \leq j < m_0$, and there are m_0 of them. That is, d can be any integer from $\mathbb{Z}_{m_0}$, and the coalition $\mathcal{S}'$ obtains no information on d.

3.1 Function Sharing Based on the Asmuth-Bloom Scheme

In [17] Kaya et al. showed how to use the Asmuth-Bloom scheme to share the private key operations of the RSA and ElGamal cryptosystems. Here we summarize their key ideas and give a brief description of their threshold RSA signature scheme.

In the original Asmuth-Bloom SSS, an iterative process was proposed to solve the system $y \equiv y_i \pmod{m_i}$. Kaya et al. [17] proposed to use a direct solution which is more suitable for function sharing. Suppose $\mathcal{S}$ is a coalition of t users gathered to construct the secret d.

1. Let $M_{\mathcal{S}\setminus\{i\}}$ denote $\prod_{j\in S, j\neq i} m_j$ 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 ith user computes

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

2. y is computed as

$$y = \sum_{i\in\mathcal{S}} u_i \bmod M_{\mathcal{S}}. \tag{2}$$

3. The secret d is computed as

$$d = y \bmod m_0.$$

As observed in [17], m_0 in the Asmuth-Bloom SSS need not to be prime and the scheme works correctly as long as m_0 is relatively prime to m_i, $1 \leq i \leq n$. Also, m_0 need not to be known during the secret construction process until the 3rd step above.

In Kaya et al.'s work [17], the RSA signature function is shared as follows: In the RSA setup, choose p, q to be safe primes, where $N = pq$, and e and d are the public and the private exponents with $ed \equiv 1 \pmod{\phi(N)}$. Share the private key d using the Asmuth-Bloom SSS with $m_0 = \phi(N)$. Given $y_i = y \bmod m_i$, each user computes $s_i = w^{u_i}$ as their partial signature. The incomplete signature $\overline{s}$ is obtained by combining the s_i values as $\overline{s} = \prod_{i\in\mathcal{S}} s_i \bmod N$, which is then converted into the actual signature s through a correction process.

4 Threshold Paillier Cryptosystem

A recent popular public key cryptosystem is a system proposed by Paillier [10] in 1999. The Paillier cryptosystem is based on properties of Carmichael function in $\mathbb{Z}_{N^2}$. Security of the cryptosystem is based on the intractability of computing discrete logarithms in $\mathbb{Z}_{N^2}$ without the Carmichael number $\lambda(N)$.

Key Generation: Let $N = pq$ where p and q are large prime integers. Let g be an arbitrary element from $\mathbb{Z}^*_{N^2}$ such that its order is a multiple of N. Let λ denote the Carmichael function of N, i.e., $\lambda = (p-1)(q-1)/2$. The public and private keys are (N, g) and λ respectively. Note that for any $x \in \mathbb{Z}^*_{N^2}$,

$$x^{\lambda} \equiv 1 \bmod N$$
$$x^{N\lambda} \equiv 1 \bmod N^2$$

Encryption: Let w be the message to be encrypted. Choose a random $r \in \mathbb{Z}^*_{N^2}$ and compute the ciphertext as $c = g^w \cdot r^N \bmod N^2$.

Decryption: One can obtain the plaintext by computing

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

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

Paillier proved that this encryption scheme is semantically secure under the assumption that it is hard to detect whether a given random element in $\mathbb{Z}^*_{N^2}$ is an N-residue.

The Paillier encryption function possesses the following homomorphic properties:

$$E(w_1 + w_2) = E(w_1) \cdot E(w_2)$$
$$E(k \cdot w) = E(w)^k$$

These properties are very useful in various multi-party protocols such as e-voting [4, 1], private information retrieval [15], and sharing of DSA signatures [16].

4.1 The Threshold Paillier Scheme

A function sharing scheme for the Paillier cryptosystem was proposed by Fouque [8] based on Shamir secret sharing. In what follows we present an alternative threshold Paillier scheme based on the Asmuth-Bloom SSS.

1. In the Paillier setup, choose two safe primes $p = 2p' + 1$ and $q = 2q' + 1$ where p' and q' are also large random primes. Set $N = pq$, $\lambda = 2p'q'$ and let β, a, b be elements randomly chosen in $\mathbb{Z}^*_N$ and set $\theta = g^{\beta\lambda} \bmod N^2$ where $g \in \mathbb{Z}^*_{N^2}$ and its order is a multiple of N. Set the public key be (N, g, θ). The secret key $\beta\lambda$ is shared with $m_0 = \lambda N$.
2. Let c be the ciphertext to be decrypted where $c = g^w r^N$ and assume a coalition $\mathcal{S}$ of size t wants to obtain the plaintext w. The ith person in the coalition knows m_j for all $j \in \mathcal{S}$ and $y_i = y \bmod m_i$ as its secret share.
3. Each user $i \in \mathcal{S}$ computes

$$u_i = y_i M'_{\mathcal{S},i} M_{\mathcal{S}\setminus\{i\}} \bmod M_{\mathcal{S}}, \tag{3}$$
$$s_i = c^{u_i} \bmod N^2. \tag{4}$$
$$\theta_i = g^{u_i} \bmod N^2 \tag{5}$$

4. The incomplete decryptor $\overline{s}$ is obtained by combining the s_i values

$$\overline{s} = \prod_{i \in \mathcal{S}} s_i \bmod N^2. \tag{6}$$

5. The θ_i values will be used to find correct value of θ which will be used to correct the incomplete decryptor. Compute the incomplete public key $\overline{\theta}$ as

$$\overline{\theta} = \prod_{i \in \mathcal{S}} \theta_i \bmod N^2. \tag{7}$$

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

$$\overline{\theta}\kappa_\theta^j \stackrel{?}{\equiv} \theta \bmod N^2 \tag{8}$$

for $0 \leq j < t$.

6. Compute the plaintext w as

$$s = \overline{s}{\kappa_s}^\delta \bmod N^2 \tag{9}$$

$$w = \frac{L(s)}{L(\theta)} \bmod N \tag{10}$$

where δ denotes the j value that satisfies (8).

The decryptor $\overline{s}$ is *incomplete* since we need to obtain $y = \sum_{i \in \mathcal{S}} u_i \bmod M_\mathcal{S}$ as the exponent of c. Once this is achieved, $c^y \equiv c^{\beta\lambda} \bmod N^2$ since $y = \beta\lambda + aN\phi(N)$ for some a.

When the equality in (8) holds we know that $\theta = g^{\beta\lambda} \bmod N^2$ 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 (3) and (5) are first reduced modulo $M_\mathcal{S}$. So, combining t of them will give $\beta\lambda + am_0 + \delta M_\mathcal{S}$ in the exponent in (7) for some $\delta \leq t - 1$. Thus in (7), we obtained

$$\overline{\theta} = g^{\beta\lambda + am_0 + \delta M_\mathcal{S}} \bmod N^2 \equiv g^{\beta\lambda + \delta M_\mathcal{S}} \equiv \theta g^{\delta M_\mathcal{S}} \equiv \theta \kappa_\theta^{-\delta} \bmod N^2$$

and for $j = \delta$ equality must hold. Note that in (7) and (8) our purpose is not to compute 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 θ.

5 Threshold Naccache-Stern Knapsack Cryptosystem

A different cryptosystem which uses bitwise encryption was proposed by Naccache and Stern [14]. This cryptosystem is based on a type of knapsack problem: Given arbitrary integers c, l, p, and a vector of integers $x = (x_1, ..., x_n)$, find a vector $w \in \{0, 1\}^l$ such that

$$c \equiv \prod_{i=1}^{l} {x_i}^{w_i} \bmod p \tag{11}$$

When the x_i are relatively prime and much smaller than the modulus p, this knapsack problem can be solved easily. When x_i are arbitrary numbers in $\mathbb{Z}_p$, the problem is hard.

Key Generation: Let p be a large prime, l is a positive integer and for i from 1 to l, set p_i to be the ith prime, starting with $p_1 = 2$. Choose a secret integer $d < p - 1$, such that $\gcd(p-1, d) = 1$. Set $v_i = \sqrt[d]{p_i} \bmod p$. The public key is then p, l, $v = (v_1, \ldots, v_l)$. The private key is d.

Encryption: To encrypt an l-bit long message w, calculate

$$c = \prod_{i=1}^{l} {v_i}^{w_i} \bmod p \tag{12}$$

where w_i is the ith bit of message w.

Decryption: One can obtain the plaintext by computing

$$w = \sum_{i=1}^{l} \frac{\gcd(p_i, c^d \bmod p) - 1}{p_i - 1} \times 2^i \tag{13}$$

5.1 The Threshold Naccache-Stern Scheme

To the best of our knowledge, no FSSs have been proposed for the Naccache-Stern knapsack cryptosystem. Here we give the first realization of an FSS for this cryptosystem with Asmuth-Bloom SSS:

1. In the Naccache-Stern Knapsack setup, choose p be a safe prime, l be a positive integer and for i from 1 to l, set p_i to be the ith prime, starting with $p_1 = 2$. Choose a secret integer $d < p - 1$, such that $\gcd(p-1, d) = 1$. Set $x_i = \sqrt[d]{p_i} \bmod p$. Set the public key be p, l, x. The private key d is shared with $m_0 = p - 1$.
2. Let c be the ciphertext to be decrypted where $c = \prod_{i=1}^{l} {x_i}^{w_i} \bmod p$ and assume a coalition $\mathcal{S}$ of size t wants to obtain the plaintext w. The ith person in the coalition knows m_j for all $j \in \mathcal{S}$ and $y_i = y \bmod m_i$ as its secret share.
3. Each user $i \in \mathcal{S}$ computes

$$u_i = y_i M'_{\mathcal{S},i} M_{\mathcal{S}\setminus\{i\}} \bmod M_{\mathcal{S}}, \tag{14}$$

$$s_i = c^{u_i} \bmod p. \tag{15}$$

4. The incomplete decryptor $\overline{s}$ is obtained by combining the s_i values

$$\overline{s} = \prod_{i \in \mathcal{S}} s_i \bmod p. \tag{16}$$

5. Let $\kappa = c^{-M_{\mathcal{S}}} \bmod p$ be the *corrector*. The corrector exponent δ can be obtained by trying

$${x_1}^{\overline{s}\kappa^j} \stackrel{?}{\equiv} 2 \bmod p \tag{17}$$

for $0 \leq j < t$.

6. Compute the plaintext message w as

$$s = \overline{s}\kappa^{\delta} \bmod p, \tag{18}$$

$$w = \sum_{i=1}^{l} \frac{(\gcd(p_i, s \bmod p) - 1)}{p_i - 1} \times 2^i \tag{19}$$

Where δ denotes the j value that satisfies (17).

The decryptor $\overline{s}$ is *incomplete* since we need to obtain $y = \sum_{i \in \mathcal{S}} u_i \bmod M_{\mathcal{S}}$ as the exponent of c. Once this is achieved, $c^y \equiv c^d \bmod p$, since $y = d + a(p-1)$ for some a.

Note that the equality in (17) must hold for one $j \leq t-1$ since the u_i values were already reduced modulo $M_{\mathcal{S}}$. So, combining t of them in (16) will give $d + am_0 + \delta M_{\mathcal{S}}$ in the exponent for some $\delta \leq t-1$. Thus we obtained

$$\overline{s} = c^{d+am_0+\delta M_{\mathcal{S}}} \equiv c^{d+\delta M_{\mathcal{S}}} \equiv sc^{\delta M_{\mathcal{S}}} \equiv s\kappa^{-\delta} \bmod p \tag{20}$$

and for $j = \delta$, equation (17) will hold.

6 Discussion of the Proposed Schemes

In this paper, we showed how the ideas of Kaya et al. [17] for function sharing with the Asmuth-Bloom secret sharing scheme can be extended to Paillier and Naccache-Stern knapsack cryptosystems and presented a FSS for the decryption operation of these cryptosystems.

The proposed schemes are efficient in terms of computational complexity. Each user needs to do $O(t)$ multiplications, one inversion, and two exponentiations for computing a partial result in the threshold Paillier scheme. Note that, in the threshold Naccache-Stern scheme, one exponentiation is sufficient for this phase. In both schemes, combining the partial results takes $t-1$ multiplications, plus possibly a correction phase which takes an exponentiation and $t-1$ multiplications.

Paillier and Naccache-Stern cryptosystems have their distinct uses due to their homomorphic properties. These homomorphic properties make the cryptosystems suitable for multi-party protocols such as e-voting [4, 1], private information retrieval [15], and sharing of DSA signatures [16]. Such applications and protocols requiring these properties can benefit greatly from the proposed function sharing schemes.

Acknowledgments

We would like to thank Murat Ak and Faruk Göloğlu for helpful discussions on the Paillier cryptosystem. We would also like to thank an anonymous referee whose detailed comments helped improving the paper considerably.

References

[1] A. Acquisti Receipt-Free Homomorphic Elections and Write-in Ballots *Technical Report CMU-ISRI-04-116*, Institute for Software Research International, Carnegie Mellon University, April 2004.

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

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

[4] O. Baudron, P. Fouque, D. Pointcheval, J. Stern, and G. Poupard Practical multi-candidate election system. In *Proc. of the twentieth annual ACM symposium on Principles of distributed computing*, pages 274–283, New York, NY, USA, 2001. ACM Press.

[5] Y. Desmedt. Some recent research aspects of threshold cryptography. In *Information Security, First International Workshop ISW '97*, number 1196 in Lecture Notes in Computer Science, pages 158–173. Springer-Verlag, 1997.

[6] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Proc. of Crypto'89*, number 435 in Lecture Notes in Computer Science, pages 307–315. Springer-Verlag, 1990.

[7] Y. Desmedt and Y. Frankel. Homomorphic zero-knowledge threshold schemes over any finite abelian group. *SIAM Journal on Discrete Mathematics*, 7(4):667–679, 1994.

[8] P-A. Fouque, G. Poupard, and J. Stern. Sharing decryption in the context of voting or lotteries. *In Proceedings of Financial Cryptography '00*, LNCS 1962, Springer-Verlag, 2000.

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

[10] P. Paillier. Public key cryptosystems based on composite residuosity classes In *Proc. of EUROCRYPT99*, LNCS 1592, pages 223–238, Springer-Verlag 1999.

[11] A. De Santis, Y. Desmedt, Y. Frankel, and M. Yung. How to share a function securely? In *Proc. of STOC94*, pages 522–533, 1994.

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

[13] V. Shoup. Practical threshold signatures. In *Proc. of EUROCRYPT 2000, Lecture Notes in Computer Science, LNCS 1807*, pages 207–220, 2000.

[14] D. Naccache and J. Stern. A new public key cryptosystem. In *EUROCRYPT97*, pages 27–36, 1997.

[15] R. Ostrovsky and W Skeith. Private Searching on Streaming Data. *Proc. of CRYPTO 2005, LNCS 3621*, pages 223–240, 2005.

[16] P. MacKenzie and M. K. Reiter. Two-party generation of DSA signatures. *International Journal of Information Security*, 2(34): pages 218-239, 2004.

[17] K. Kaya, A. A. Selçuk and Z. Tezcan. Threshold Cryptography Based on Asmuth-Bloom Secret Sharing. *Proceedings of ISCIS'06*, Lecture Notes in Computer Science, Springer-Verlag, 2006.