Practical Threshold Signatures with Linear Secret Sharing Schemes*

İlker Nadi Bozkurt, Kamer Kaya**, and Ali Aydın Selçuk

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

Abstract. Function sharing deals with the problem of distribution of the computation of a function (such as decryption or signature) among several parties. The necessary values for the computation are distributed to the participating parties using a secret sharing scheme (SSS). Several function sharing schemes have been proposed in the literature, with most of them using Shamir secret sharing as the underlying SSS. In this paper, we investigate how threshold cryptography can be conducted with any linear secret sharing scheme and present a function sharing scheme for the RSA cryptosystem. The challenge is that constructing the secret in a linear SSS requires the solution of a linear system, which normally involves computing inverses, while computing an inverse modulo $\varphi(N)$ cannot be tolerated in a threshold RSA system in any way. The threshold RSA scheme we propose is a generalization of Shoup's Shamir-based scheme. It is similarly robust and provably secure under the static adversary model. At the end of the paper, we show how this scheme can be extended to other public key cryptosystems and give an example on the Paillier cryptosystem.

Keywords: Linear secret sharing, threshold cryptography, function sharing.

1 Introduction

The secure storage of the private keys of a cryptosystem is an important problem. Possession of a highly sensitive key by an individual may not be desirable as the key can easily be lost or as the individual may not be fully trusted. Giving copies of the key to more than one individual increases the risk of compromise. A solution to this problem is to give shares of the key to several individuals, forcing them to cooperate to find the secret key. This not only reduces the risk of losing the key but also makes compromising the key more difficult. In threshold

* This work is supported in part by the Turkish Scientific and Technological Research Agency (TÜBİTAK), under grant number 108E150.

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

B. Preneel (Ed.): AFRICACRYPT 2009, LNCS 5580, pp. 167–178, 2009.

cryptography, secret sharing deals with this problem, namely, sharing a highly sensitive secret among a group of n users such that only when a sufficient number t of them come together can the secret be reconstructed. Well-known secret sharing schemes (SSS) in the literature include Shamir [17] based on polynomial interpolation, Blakley [2] based on hyperplane geometry, and Asmuth-Bloom [1] based on the Chinese Remainder Theorem.

A shortcoming of secret sharing schemes is the need to reveal the secret shares during the reconstruction phase. The system would be more secure if the subject function can be computed without revealing the secret shares or reconstructing the secret. This is known as the function sharing problem. A function sharing scheme 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 have been proposed in the literature [4,5,6,7,18,10,16]. Nearly all these protocols use Shamir secret sharing as the underlying SSS.

1.1 Secret Sharing Schemes

The problem of secret sharing and the first solutions were introduced in 1979 independently by Shamir [17] and Blakley [2]. 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.

Shamir secret sharing is based on polynomial interpolation over a finite field. It uses the fact that we can find a secret polynomial of degree $t-1$ given t data points. To generate a polynomial $f(x) = \sum_{i=0}^{t-1} a_i x^i$, a_0 is set to the secret value and the coefficients a_1 to a_{t-1} are assigned random values in the field. The value $f(i)$ is given to user i. When t out of n users come together, they can construct the polynomial using Lagrange interpolation and can find the secret.

Blakley secret sharing scheme has a different approach based on hyperplane geometry: To implement a (t, n) threshold scheme, each of the n users is given a hyperplane equation in a t dimensional space over a finite field such that each hyperplane passes through a certain point. The intersection point of the hyperplanes is the secret. When t users come together, they can solve the system of equations to find the secret.

Both Shamir and Blakley are linear threshold secret sharing schemes: As Karnin et al. [11] observed, Shamir SSS is a subclass of a broader class of linear secret sharing. The polynomial share computation can be represented as a matrix multiplication by using a Vandermonde matrix. Similarly, the secret and the shares of the Blakley SSS can be represented as a linear system $Ax = y$ where the matrix A and the vector y are obtained from the hyperplane equations.

1.2 Function Sharing Schemes

Function sharing is the concept of distributing the computation of a function such that when a sufficient number of users come together they can compute

the value of the function without revealing their secret shares. This problem is related to secret sharing as the secret values needed for partial computations are distributed using secret sharing.

Several solutions for sharing the RSA, ElGamal, and Paillier private key operations have been proposed in the literature [4,5,6,7,9,12,13,16,18]. Almost all of these schemes have been based on the Shamir SSS.

The additive nature of the Lagrange's interpolation formula used in the combining phase of Shamir's scheme makes it an attractive choice for function sharing, but it also provides several challenges. One of the most significant challenges is the computation of inverses in $\mathbb{Z}_{\varphi(N)}$ for the division operations in Lagrange's formula, while $\varphi(N)$ should not be known by the users. There are two main difficulties in this respect:

1. An inverse x^{-1} will not exist modulo $\varphi(N)$ if $\gcd(x, \varphi(N)) \neq 1$.
2. Even when x^{-1} exists it should not be computable by a user, since that would enable computing $\varphi(N)$.

Early solutions to this problem, albeit not very efficient, were given in [4,16]. Afterwards an ingenious solution was given by Shoup [18] where he removed the need of taking inverses in Lagrange interpolation altogether.

Shoup's practical RSA scheme has inspired similar works on different cryptosystems. Fouque et al. [9] proposed a similar Shamir-based threshold solution for the Paillier cryptosystem and used it in e-voting and lottery protocols. Later, Lysyanskaya and Peikert [13] improved this work and obtained a threshold Paillier encryption scheme secure under the adaptive security model. The current paper is also inspired by Shoup's work.

1.3 Our Contribution

In this work, we show how to generalize Shoup's ideas to do function sharing with any linear SSS, and we give a robust threshold RSA signature scheme. A linear SSS, where the solution is based on solving a linear system, naturally requires computing inverses for reconstructing the secret. We show how to utilize such a system for function sharing while avoiding computation of inverses modulo $\varphi(N)$ completely.

We also discuss how this approach can be applied to other public key cryptosystems and show an example on the Paillier decryption function.

2 Linear Secret Sharing Schemes

A linear (t, n) threshold secret sharing scheme can be defined as follows: Let A be a full-rank public $n \times t$ matrix with entries chosen from $\mathcal{F} = \mathbb{Z}_m^*$ for a prime m. Let $x = (x_1, x_2, \ldots, x_t)^T$ be a secret vector from $\mathcal{F}^t$. Let a_{ij} denote the entry at the ith row and jth column of the matrix A.

2.1 Dealing Phase

The dealer chooses a secret vector $x \in \mathcal{F}^t$ where the first entry x_1 is set to the secret value (the RSA private key d in our case) and the values of the other coordinates are set randomly from the field $\mathcal{F}$. The ith user will get a his share $y_i \in \mathcal{F}$,

$$y_i = a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{it}x_t. \tag{1}$$

For a (t, n) threshold scheme there will be n such shares, and hence we will have an $n \times t$ linear system

$$Ax = y. \tag{2}$$

The dealer then sends the secret value of y_i to user i for $1 \leq i \leq n$ and makes the matrix A public.

2.2 Share Combining Phase

Share combining step is simply finding the solution of a linear system of equations. Suppose that a coalition $\mathcal{S} = \{i_1, \ldots, i_t\}$ of users come together. They form a matrix $A_{\mathcal{S}}$ using their equations and solve

$$A_{\mathcal{S}}x = y_{\mathcal{S}}, \tag{3}$$

where $y_{\mathcal{S}}$ is the vector of the secret shares of the users. The secret is found as the first coordinate of the solution.

3 Sharing RSA Signature Computation

In this section, we describe our threshold RSA signature scheme which works with any linear SSS in general.

3.1 Setup

In the RSA setup phase, choose the RSA primes $p = 2p' + 1$ and $q = 2q' + 1$, where p' and q' are large primes. Compute the RSA modulus as $N = pq$. Let $m = p'q'$. The public key e is chosen as a prime number, details of which will be explained in the next section. After choosing e, the private key d is computed such that $ed \equiv 1 \pmod{m}$. Then the dealer shares the private key d among n users using a linear threshold SSS described in Section 2.

The dealer also chooses v as a generator of Q_N, where Q_N is the subgroup of squares in $\mathbb{Z}_N^*$. He computes and broadcasts

$$v_i = v^{y_i} \in Q_N, \tag{4}$$

for $1 \leq i \leq n$, which are the verification keys to be used in the proofs of correctness of the partial signatures, where y_i is the secret share of user i.

3.2 Signing

Let $H(.)$ be a hash function mapping input messages to $\mathbb{Z}_N^*$ and let $w = H(M) \in \mathbb{Z}_N^*$ be the hashed message to be signed. Assume a coalition $\mathcal{S}$ of size t wants to obtain the signature $s = w^d \bmod N$.

Generating partial signatures. Let $\mathcal{S} = \{i_1, \ldots, i_t\}$ be the coalition of t users, forming the linear system

$$A_{\mathcal{S}} x = y_{\mathcal{S}}.$$

Let c_{ij} be the ij-th cofactor of matrix $A_{\mathcal{S}}$ and let $C_{\mathcal{S}}$ be the adjugate matrix,

$$C_{\mathcal{S}} = \begin{pmatrix} c_{11} & c_{21} & \ldots & c_{t1} \\ c_{12} & c_{22} & \ldots & c_{t2} \\ \vdots & \vdots & \ddots & \vdots \\ c_{1t} & c_{2t} & \ldots & c_{tt} \end{pmatrix}.$$

If we denote the determinant of $A_{\mathcal{S}}$ by $\Delta_{\mathcal{S}}$ we have,

$$A_{\mathcal{S}} C_{\mathcal{S}} = C_{\mathcal{S}} A_{\mathcal{S}} = \Delta_{\mathcal{S}} I_t, \tag{5}$$

where I_t denotes the $t \times t$ identity matrix.

For our scheme, each user $i \in \mathcal{S}$ computes his partial signature as

$$s_i = w^{2c_{i1} y_i} \bmod N. \tag{6}$$

Verifying partial signatures. Each user computes and publishes a proof of correctness for the verification of his partial signature. The proof of correctness of the partial signature of user i is a proof that the discrete logarithm of s_i^2 to the base

$$\tilde{s}_i = w^{4c_{i1}} \bmod N \tag{7}$$

is the same as the discrete logarithm of v_i to the base v. To prove this, a protocol by Shoup [18] which is a non-interactive version of Chaum and Pedersen's [3] interactive protocol is used:

Let $L(n)$ be the bit-length of n. Let H' be a hash function, whose output is an L_1-bit integer, where L_1 is a secondary security parameter. To construct the proof of correctness, user i chooses a random number $r \in \{0, 1, \ldots, 2^{L(N)+2L_1} - 1\}$, computes

$$\begin{aligned} v' &= v^r \bmod N, \\ s' &= \tilde{s}_i^r \bmod N, \\ D &= H'(v, \tilde{s}_i, v_i, s_i^2, v', s'), \\ \sigma &= y_i D + r. \end{aligned}$$

Then user i publishes his proof of correctness as (σ, D).

To verify this proof of correctness, one checks whether

$$D \stackrel{?}{=} H'(v, \tilde{s}, v_i, s_i^2, v^\sigma v_i^{-D}, \tilde{s}_i^\sigma s_i^{-2D}).$$

Combining partial signatures. To combine the partial signatures, we simply compute

$$\overline{s} = \prod_{i \in \mathcal{S}} s_i \bmod N. \tag{8}$$

Note that, by equation (5), we have

$$\overline{s} = w^{d\,\delta} \bmod N, \tag{9}$$

where

$$\delta = 2\,\Delta_{\mathcal{S}}. \tag{10}$$

Given that e is a prime number relatively prime to $\Delta_{\mathcal{S}}$, it is easy to compute the signature $s = w^d \bmod N$ from $\overline{s}$. Take

$$s = \overline{s}^a w^b \bmod N, \tag{11}$$

where a and b are integers such that

$$\delta a + eb = 1, \tag{12}$$

which can be obtained by the extended Euclidean algorithm on δ and e.

4 Choosing e

The choice of e is critical in the setup phase because the solution depends on e and $\Delta_{\mathcal{S}}$ being relatively prime. To achieve this, we can either choose a special matrix whose determinant is known to be relatively prime to e, or choose e as a sufficiently large prime according to t and n so that the probability that $\Delta_{\mathcal{S}}$ is divisible by e will be negligible for any coalition $\mathcal{S}$.

4.1 Choosing e probabilistically

The probability of a random integer's being divisible by a prime e is $1/e$. So, if we have a (t, n) threshold scheme, the probability that the determinant of none of the $\binom{n}{t}$ $A_{\mathcal{S}}$ matrices will be divisible by e is $\left(1 - \frac{1}{e}\right)^{\binom{n}{t}}$. If we take $e \gg \binom{n}{t}$, we have

$$\left(1 - \frac{1}{e}\right)^{\binom{n}{t}} \approx 1\ . \tag{13}$$

4.2 Choosing a Vandermonde Matrix as the Coefficient Matrix

A simple choice for the matrix A that enables us to guarantee that e will be relatively prime to the determinant of the coefficient matrix is to choose the rows of the matrix A as the rows of a Vandermonde matrix. Note that this is exactly the case for Shamir secret sharing. Then $A_{\mathcal{S}}$ will have the following form for a coalition $\mathcal{S}$ of size t:

$$A_{\mathcal{S}} = \begin{pmatrix} 1 & a_1 & a_1^2 & \dots & a_1^{t-1} \\ 1 & a_2 & a_2^2 & \dots & a_2^{t-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & a_t & a_t^2 & \dots & a_t^{t-1} \end{pmatrix}$$

The determinant of the Vandermonde matrix is nonzero, provided that no two rows are identical, and is given by the following formula:

$$|A_{\mathcal{S}}| = \prod_{i,j=1, i<j}^{t} (a_i - a_j) \tag{14}$$

Without loss of generality take $(a_1, a_2, \dots, a_n) = (1, 2, \dots, n)$. Obviously,

$$\prod_{i,j=1,i<j}^{t} (a_i - a_j) \;\mid\; \prod_{i,j=1,i<j}^{n} (a_i - a_j).$$

We also have,

$$\prod_{i,j=1,i<j}^{n} (a_i - a_j) = 1^{\alpha_1} 2^{\alpha_2} \dots (n-1)^{\alpha_{n-1}} \tag{15}$$

for some $\alpha_1, \alpha_2, \dots, \alpha_{n-1}$. Hence by choosing e as a prime greater than or equal to n we can guarantee that the determinant of any $A_{\mathcal{S}}$ will be relatively prime to e.

5 Security Analysis

Now we will prove that the proposed threshold RSA signature scheme is secure provided that the standard RSA signature is secure. We assume a static adversary model in the sense that the adversary controls exactly $t-1$ users and chooses them at the beginning of the attack. The adversary obtains all secret information of the corrupted users along with the public parameters of the system. She can control the actions of the corrupted users, asking for partial signatures of messages of her choice but cannot corrupt any other user in due course.

First we will analyze the proof of correctness. Then using this analysis we will prove that the proposed threshold signature scheme is secure.

5.1 Analysis of the Proof of Correctness

For generating and verifying the proof of correctness, the following properties hold:

Completeness. If the ith user is honest then the proof succeeds since

$$v^{\sigma} v_i^{-D} = v^{y_i D} v^r v_i^{-D} = v^r = v'$$

and

$$\tilde{s}_i^{\sigma} s_i^{-2D} = w^{4c_{i1}(y_i D + r)} w^{-4c_{i1} y_i D} = s^r = s'.$$

Soundness. To prove the soundness of the proof of correctness, we have to show that the adversary cannot construct a valid proof of correctness for an incorrect share, except with negligible probability. Let (σ, D) be a valid proof of correctness for a message w and partial signature s_i. We have $D = H'(v, \tilde{s}_i, v_i, s_i^2, v', s')$, where

$$\tilde{s}_i = w^{4c_{i1}}, v' = v^{\sigma} v_i^{-D}, s' = \tilde{s}_i^{\sigma} s_i^{-2D}.$$

Obviously $\tilde{s}_i$, v_i, s_i^2, v' and s' all lie in Q_n and we know that v is a generator of Q_n. So we have

$$\tilde{s}_i = v^{\alpha}, v_i = v^{y_i}, s_i^2 = v^{\beta}, v^{'} = v^{\gamma}, s' = v^{\mu},$$

for some integers $\alpha, \beta, \gamma, \mu$. From this we have,

$$\sigma - Dy_i \equiv \gamma \pmod{m} \tag{16}$$

$$\sigma\alpha - D\beta \equiv \mu \pmod{m}. \tag{17}$$

From equations (16) and (17) we get,

$$D(\beta - y_i\alpha) \equiv \alpha\gamma - \mu \pmod{m}. \tag{18}$$

A share is correct, if and only if,

$$\beta \equiv y_i\alpha \pmod{m}. \tag{19}$$

If (19) does not hold, then it does not hold either mod p' or mod q' and so (18) uniquely determines D mod p' or D mod q'. But the distribution of D is uniform in the random oracle model, so this happens with negligible probability.

Zero knowledge simulatability. To prove zero knowledge simulatability, we will use the random oracle model for the hash function and construct a simple simulator that simulates the adversary's view without knowing the value y_i. When an uncorrupted user wants to create a proof (σ, D) for a message w and partial signature s_i, the simulator chooses $D \in \{0, \ldots, 2^{L_1} - 1\}$ and $\sigma \in \{0, \ldots, 2^{L(N)+2L_1} - 1\}$ at random and defines the value of the random oracle at $(v, \tilde{s}_i, v_i, s_i^2, v^{\sigma} v_i^{-D}, \tilde{s}_i^{\sigma} s_i^{-2D})$ to be D. Note that, the value of the random oracle is not defined at this point with all but negligible probability. When the adversary queries the oracle, if the value of the oracle was already set the simulator returns that value, otherwise it returns a random value. It is obvious that the output of this simulator is statistically indistinguishable from real output.

5.2 Security of the Proposed Signature Scheme

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

Theorem 1. *In the random oracle model for H', the proposed threshold signature scheme is a secure threshold signature scheme (robust and non-forgeable) under the static adversary model given that the standard RSA signature scheme is secure.*

Proof. We will simulate the threshold protocol with no information on the secret where the output of the simulator is indistinguishable in the adversary's 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 signature scheme.

Let $i_1, \ldots, i_{t-1}$ be the set of corrupted players. To simulate the adversary's view, we simply choose the y_{i_j} values belonging to the set of corrupted players at random from the set $\{0, \ldots, \lfloor N/4 \rfloor - 1\}$. The corrupted players' secret key shares are random numbers in the set $\{0, \ldots, m-1\}$. Once these values are chosen, the values y_i for the uncorrupted players are completely determined modulo m, but cannot easily be computed. However, given $w, s \in \mathbb{Z}_N^*$ with $s^e = w$, we can easily compute s_{i_t} for an uncorrupted user i_t as

$$s_{i_t} = w^{2c_{t1}y_{i_t}} = s^{2\Delta_S} w^{-2\sum_{j=1}^{t-1} c_{j1}y_{i_j}}. \tag{20}$$

Note the dependence of Δ_S and c_{j1} values on the coalition $\{i_1, \ldots, i_{t-1}, i_t\}$.

Using this technique, we can generate the values $v, v_1, \ldots, v_n$, and also generate any share s_i of a signature, given the standard RSA signature. These values produced by the simulator and the proof of correctness given in this section are computationally indistinguishable from the real ones. Hence, the threshold RSA signature scheme based on a linear SSS is secure given that the standard RSA signature scheme is secure. □

6 Application to Other PKCs

So far, we investigated only how to share the RSA signature function by using a linear SSS. The same approach can also be used to share the RSA decryption function since the signature and decryption functions are mostly identical. Besides RSA, the proposed approach can also be used to share other public key cryptosystems where the private key is used in the exponent, such as the ElGamal [8], Naccache-Stern [14] and the Paillier [15] decryption functions.

Below, as an example, we describe how our approach can be utilized for sharing the Paillier decryption function. The scheme works along the same lines as Fouque et al.'s extension [9] of Shoup's work to the Paillier cryptosystem.

6.1 The Paillier Cryptosystem

The Paillier PKC is based on the properties of Carmichael function over $\mathbb{Z}_{N^2}$ where N is an RSA composite. 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 $\lambda = (p-1)(q-1)/2$ denote the the Carmichael function for N. The public and private keys are (N, g) and λ, respectively.

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

$$s = g^w r^N \bmod N^2.$$

Decryption. The plaintext is obtained by

$$w = \frac{L(s^\lambda \bmod \mathbb{Z}_{N^2})}{L(g^\lambda \bmod \mathbb{Z}_{N^2})} \bmod N$$

where $L(x) = (x-1)/N$ for $x \equiv 1 \pmod N$.

Paillier proved that this 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 cryptosystem possesses the following homomorphic properties:

$$E(w_1 + w_2) = E(w_1).E(w_2)$$
$$E(k.w) = E(w)^k.$$

6.2 Sharing the Paillier Decryption Function

Since $\lambda(N)$ must be kept secret, the inverse computation problem is similar to the one we encountered while sharing the RSA signature function. Our threshold Paillier scheme is given below:

Key Generation. In the Paillier setup phase, choose two safe primes $p = 2p'+1$ and $q = 2q'+1$, where p' and q' are large primes and $\gcd(N, \varphi(N)) = 1$ for $N = pq$. Let $m = p'q'$. Let $\beta \in_R \mathbb{Z}_N^*$ and $(a, b) \in_R \mathbb{Z}_N \times \mathbb{Z}_N^*$. Compute

$$g = (1+N)^a \times b^N \bmod N^2.$$

Share the private key $d = \beta m$ among n users with modulo Nm by using the linear SSS. Let

$$\theta = L\left(g^{\beta m}\right) = a\beta m \bmod N.$$

Set the public key as (g, N, θ). Choose v as a generator of Q_{N^2}, where Q_{N^2} is the cyclic group of squares in $\mathbb{Z}_{N^2}$. Compute the verification keys

$$v_i = v^{y_i} \in Q_{N^2}$$

for $1 \le i \le n$ as before.

Encryption. Let w be the message to be encrypted. Choose a random $r \in \mathbb{Z}_{N^2}$ and compute the ciphertext as $s = g^w r^N \bmod N^2$. Let $m = p'q'$.

Decryption. Let s be the ciphertext to be decrypted and $\mathcal{S} = \{i_1, \ldots, i_t\}$ denote a coalition of t users that will compute the plaintext together. Let $A_{\mathcal{S}}$ be the coalition matrix and $C_{\mathcal{S}}$ be the corresponding adjugate matrix, respectively, as in Section 3. Each member $i \in \mathcal{S}$ computes his partial value as

$$s_i = s^{2c_{i1}y_i} \bmod N^2$$

where c_{i1} is the ith element of the first row of $C_{\mathcal{S}}$. He also generates a proof of correctness which is used to prove that the discrete logarithm of s_i^2 to the base $\tilde{s} = w^{4c_{i1}}$ is the same as the discrete logarithm of v_i to the base v. Note that the proof is now working on a cyclic group of unknown order mN.

After the partial decryptions are obtained, the combining algorithm computes the plaintext

$$w = \frac{L\left(\prod_{i\in\mathcal{S}} s_i \bmod N^2\right)}{2\Delta_{\mathcal{S}}\theta} \bmod N.$$

Note that

$$\begin{aligned}\prod_{i\in\mathcal{S}} s_i &\equiv s^{2\Delta_{\mathcal{S}}\beta m} \\ &\equiv g^{2\Delta_{\mathcal{S}}\beta m w} \\ &\equiv (1+N)^{2\Delta_{\mathcal{S}} a\beta m w} \\ &\equiv 1 + 2\Delta_{\mathcal{S}} a\beta m w N \\ &\equiv 1 + 2\Delta_{\mathcal{S}}\theta w N \pmod{N^2}.\end{aligned}$$

7 Conclusion

We showed how to do threshold cryptography with linear secret sharing in general. We presented a robust RSA threshold signature scheme based on a linear SSS. The proposed signature scheme generalizes Shoup's threshold RSA signature based on Shamir secret sharing, and is as efficient and practical as Shoup's scheme.

Besides RSA, this approach can be extended to other public key cryptosystems where the private key is used in the exponent. As an example we demonstrated how Paillier decryption function can be shared by this approach. ElGamal and Naccache-Stern knapsack cryptosystems are some other systems that can benefit from the proposed solution.

Acknowledgements

We would like to thank Ahmet Güloğlu for informative discussions and his comments on this paper. We would also like to thank anonymous AfricaCrypt referees for their valuable comments which significantly helped to improve the paper.

References

1. Asmuth, C., Bloom, J.: A modular approach to key safeguarding. IEEE Trans. Information Theory 29(2), 208–210 (1983)
2. Blakley, G.: Safeguarding cryptographic keys. In: Proc. of AFIPS National Computer Conference (1979)
3. 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)
4. Desmedt, Y.: Some recent research aspects of threshold cryptography. In: Okamoto, E. (ed.) ISW 1997. LNCS, vol. 1396, pp. 158–173. Springer, Heidelberg (1998)
5. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 307–315. Springer, Heidelberg (1990)
6. 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)
7. Desmedt, Y., Frankel, Y.: Homomorphic zero-knowledge threshold schemes over any finite abelian group. SIAM Journal on Discrete Mathematics 7(4), 667–679 (1994)
8. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Trans. Information Theory 31(4), 469–472 (1985)
9. Fouque, P.A., Poupard, G., Stern, J.: Sharing decryption in the context of voting or lotteries. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 90–104. Springer, Heidelberg (2001)
10. Huang, H.F., Chang, C.C.: A novel efficient (t,n) threshold proxy signature scheme. Information Sciences 176(10), 1338–1349 (2006)
11. Karnin, E.D., Greene, J.W., Hellman, M.E.: On secret sharing systems. IEEE Transactions on Information Theory 29, 35–41 (1983)
12. Kaya, K., Selçuk, A.A.: Threshold cryptography based on Asmuth–Bloom secret sharing. Information Sciences 177(19), 4148–4160 (2007)
13. 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)
14. Naccache, D., Stern, J.: A new public key cryptosystem. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 27–36. Springer, Heidelberg (1997)
15. 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)
16. De Santis, A., Desmedt, Y., Frankel, Y., Yung, M.: How to share a function securely? In: Proc. of STOC 1994, pp. 522–533 (1994)
17. Shamir, A.: How to share a secret? Comm. ACM 22(11), 612–613 (1979)
18. Shoup, V.: Practical threshold signatures. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 207–220. Springer, Heidelberg (2000)