Guaranteed Correct Sharing of Integer Factorization with Off-Line Shareholders

Wenbo Mao

Hewlett-Packard Laboratories
Filton Road
Stoke Gifford
Bristol BS12 6QZ
United Kingdom
wm@hplb.hpl.hp.com

Abstract. A fair public-key cryptosystem consists of multi-party protocols in which a plural number of participants (shareholders) are involved in receiving and verifying distributed shares. It will be desirable if multi-party protocols can be streamlined into two-party ones without lowering the quality of fairness: secret is still shared among many (more than two) parties. In this paper we propose a scheme that distributes secret shares of the factorization of an integer to multi-parties without their participation in the protocols for share distribution and verification. A single verifier suffices to verify the correctness of the shares using the public keys of the off-line shareholders. Due to the universal verifiability, a guaranteed correctness of secret sharing is achieved without relying on the honesty of the verifier.

1 Introduction

A number of fair cryptosystems have been proposed (e.g., [1, 11, 12, 13, 15], a good reference is in [5]). Here, fairness essentially means to share a secret among a plural number of participants called shareholders. This can be achieved via verifiable secret sharing (VSS, e.g., [1, 13, 15]) in which a dealer distributes structured messages called shares to the shareholders via confidential channels. Each of the shareholders must verify the correctness of the share received in order to be sure that whenever needed they can correctly reconstruct the secret via distributed computing.

VSS schemes are multi-party protocols which require the shareholders stay on-line to receive and verify messages. This is so even for non-interactive VSS schemes such as Pedersen's [16]. Moreover, because each shareholder verifies data in private, these schemes require the shareholders honest. Considering possible presence of dishonest shareholder(s), the previous VSS protocols resort to what we call "front-end (t, n)-threshold secret sharing" ([16, 22]), in which the secret is distributed in such a way that it can be correctly re-constructed if t $(< n)$ out of n shareholders are honest. The multi-party protocols for threshold secret sharing are very inefficient, and give the user little freedom to choose shareholders they trust (since the shareholders must pre-negotiate some system parameters,

e.g., (t, n), in order to correctly run protocols). Some schemes even resort to broadcasting channels which are impractical to realize. Still, they do not offer absolute guarantee on correctness in secret sharing.

In this paper we propose a publicly verifiable secret sharing scheme for establishing correct sharing of the factorization of an integer. The integer factorization problem forms an important class of cryptosystems and protocols such as RSA [20], Rabin [21], Fiat-Shamir [8] and Guillou-Quisquater [19]. The public verifiability means that the secret sharing need not use a plural number of on-line participating verifiers. A single verifier will suffice for the verification job, and because the verification will only use public data such as the public keys of the un-participating shareholders, the verifier can be anybody and the verification can be repeated. Thus, the new scheme not only streamlines multi-party protocols into two-party ones, but also achieves a guaranteed correctness in secret sharing.

1.1 Outline of the Paper

In the next subsection we review the previous integer factorization based fair public-key cryptosystems. In Section 2 we introduce cryptographic primitives that will be used in our technique. In Section 3 we present a new integer factorization based fair cryptosystem. Finally, we conclude the work in Section 4.

1.2 Previous Work on Integer Factorization Based Fair Cryptosystems

Micali [13] and Okamoto [15] have proposed two fair public-key cryptosystems based on the integer factorization problem.

Micali's scheme is based on the following number theoretic facts: (i) there exists efficient protocols for showing probabilistic evidence that a composite number N is the product of two primes [9, 13]; (ii) for such a number, a quadratic residue in Z_N^* has distinct square roots with opposite Jacobi symbols, and such a pair of square roots provide an efficient algorithm to factor N; and (iii) the product of several quadratic residues is itself a quadratic residue with the Jacobi symbol as the product of those of the individuals.

Using these facts, the factorization of an integer N (e.g., an RSA modulus) can be fairly shared as follows. The user generates a quadratic residue modulo N which is the product of several other quadratic residues; sends a square root with Jacobi symbol -1 of the product to a key-management center (KMC); sends a square root with Jacobi symbol 1 of each individual quadratic residue to a respective shareholder; each of the latter verifies that the value received has the correct Jacobi symbol. Thus, when all of the shareholders disclose the square roots received, N can be factored because two square roots of the same quadratic residue with opposite Jacobi symbols are now available. Obviously, the scheme is a multi-party protocol and requires each shareholder to stay on-line and honestly check the correctness of the square root received.

Okamoto's schemes (there are three of them) are based on combining bit-commitment protocols used in his electronic cash scheme [14] and Pedersen's non-interactive threshold verifiable secret sharing (VSS) [16]. So the schemes require on-line availability of multi shareholders. In two of the three schemes, the shareholders must also prepare for some session parameters for each instance of secret sharing, which are are functions of the integer that the user wants to establish as part of the public key. These parameters have to be agreed through real-time negotiations among the shareholders.

2 Cryptographic Primitives

In this section we introduce cryptographic primitives that will be used in our technique.

2.1 Notation and System Setup

Let Z denote the ring of integers. For positive integer p, let Z_p denote the ring of integers modulo p and Z_p^* denote the multiplicative group modulo p. For S being a set, $a \in_R S$ means choosing a from S at random according to the uniform distribution. For integers a, b, we write $a|b$ if a divides b and $a \not| b$ if otherwise. Let $|a|$ denote the bit length of a, and $abs(a)$, the absolute value of a. Finally, we assume a collision resistant hash function $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^l$ ($l \approx 160$).

Throughout the paper we will use two public cyclic groups constructed as follows. Let r be a large prime such that $q = 2r+1$ and $p = kq+1$ are also prime where k is an even number. Number theoretic research [10] has shown that it is not difficult to find such primes. Let $h \in Z_q^*$, $g \in Z_p^*$ be elements of order r and q, respectively, and set $H = \langle h \rangle$, $G = \langle g \rangle$ with multiplication as the respective group operations. With these settings the following two congruences hold

$$h^r = 1 (\mathrm{mod} q) \quad \text{and} \quad g^q = 1 (\mathrm{mod} p). \tag{1}$$

We assume that it is difficult to compute discrete logarithms in Z_q^* and Z_p^* to the bases h and g, respectively.

The system will also setup a fixed element $f \in G$ such that nobody knows $\log_g(f)$. Such an element can be chosen by a trusted center who sets f using a secure pseudo-random number generator which is seeded by g, and publishes this relationship between the two quantities.

2.2 Cryptosystem

Encryption will use the ElGamal cryptosystems [7]. Let $x \in Z_r$ be the private key of someone other than the verifier and

$$y = h^x \bmod q \tag{2}$$

be the matching public key. To encrypt a message $M \in Z_q$ under the public key y, the sender (Alice) chooses a session key $k \in_R Z_r$, and calculates the following pair

$$(h^k,\ y^{-k}M)(\mathrm{mod}\ q). \tag{3}$$

The ciphertext pair (A, B) can be decrypted using the private key x as follows

$$M = A^x B \bmod q. \tag{4}$$

Using well-known techniques for building robust threshold cryptosystems [6, 16, 17], the ElGamal cryptosystem can be setup as a robust threshold one. Encryption using such a cryptosystem is exactly the same as that in normal ElGamal. Decryption will be via distributed computing in a threshold manner among a set of principals who share a private decryption key matching the public key y. However, there will be no need to re-construct the shared private key in order to perform decryption. In fact, the private key has never been, and will never be constructed. Section 2.3 of [4] provides a succinct description on how to setup such a threshold ElGamal cryptosystem.

2.3 Verifiable Encryption of Discrete Log

The scheme, due to Stadler [24], will make use of double exponentiation. By double exponentiation to bases g and h we mean the following organization of the quantities $z \in Z_r$, $h \in Z_q^*$ and $g \in Z_p^*$:

$$g^{h^x \bmod q} \bmod p. \tag{5}$$

With the double exponentiation structure, we will be able to prove the following structured knowledge statement. For public quantities A, h, y, U, g,

$$\exists k:\ A = h^k(\mathrm{mod}\ q)\ \wedge\ U = g^{y^k(\mathrm{mod}\ q)}(\mathrm{mod}\ p). \tag{6}$$

Assuming with this statement proven, given value $V \in G$, and a pair of ElGamal ciphertext (A, B), one can be convinced that (A, B) encrypts $\log_g(V)$ under the public key y if $U^B = V(\mathrm{mod}\ p)$.

Using a cryptographicly secure hash function, the following procedure provides a non-interactive proof of the structure (6). For $i = 1, \cdots, l$, A prover (Alice) chooses $u_i \in_R Z_r$ and calculates

$$t_{hi} = h^{u_i} \bmod q,\quad t_{gi} = g^{(y^{u_i} \bmod q)} \bmod p. \tag{7}$$

Then she computes the following ℓ-tuples

$$R = (R_1, R_2, \cdots, R_\ell) = (u_1 - kC_1, u_2 - kC_2, \cdots, u_\ell - kC_\ell)\ (\mathrm{mod}\ r) \tag{8}$$

where C_i is the i-th bit of

$$C = \mathcal{H}(A, h, y, U, g, t_{h1}, t_{g1}, \cdots, t_{h\ell}, t_{g\ell}). \tag{9}$$

The non-interactive proof consists of pair (R, C). A verifier (Bob) verifies by computing (for $i = 1, \cdots, \ell$)

$$t_{hi} = h^{R_i} A^{C_i} \bmod q, \tag{10}$$

$$t_{gi} = (g^{(1-C_i)} U^{C_i})^{(y^{R_i})} \bmod p \tag{11}$$

and checking whether the equation (9) holds.

3 Sharing the Factorization of an Integer

In this section we assume that

$$Y = h^X \bmod q \tag{12}$$

is a public key where its discrete logarithm X has been fairly shared among a number of shareholders as in the case of the robust cryptosystem described in Section 2.2. We will specify protocols that let a user use this public key to verifiably encrypt the factorization of an integer.

We begin with introducing two observations.

The first observation can be described as follows. Let

$$(A_1 = h^{k_1},\ \ B_1 = Y^{-k_1} n_1)\ \ (\bmod q), \tag{13}$$

and

$$(A_2 = h^{k_2},\ \ B_2 = Y^{-k_2} n_2)\ \ (\bmod q) \tag{14}$$

be two pairs of ciphertext encrypting numbers n_1 and n_2 respectively under the public key Y. Multiplying A_1 to A_2, and B_1 to B_2, we get

$$(A_3 = h^{k_1+k_2},\ \ B_3 = Y^{-(k_1+k_2)} n_1 n_2)\ \ (\bmod q). \tag{15}$$

The sender can disclose $n_1 n_2 (\bmod q)$ by revealing $k_1 + k_2 (\bmod r)$ since

$$n_1 n_2 = Y^{k_1+k_2} B_1 B_2 (\bmod q). \tag{16}$$

The value $n_1 n_2 (\bmod q)$ opened from (16) will not reveal any useful information about n_1 or n_2 since arbitrarily setting $n_1 \leq q$, there exists an $n_2 \leq q$ with $n_1 n_2 (\bmod q)$ meeting the left-hand side of (16).

If we have a method to decide the bit-lengths of the messages encrypted in (13) and (14), then under certain conditions we can decide whether the quantities encrypted will precisely divide the value opened. The needed conditions form our second observation which are stated in the following lemma.

Lemma *Let* $n_1 n_2 = n (\bmod q)$ *and* $|n| + 2 < |q|$. *If* $|n_1| + |n_2| \leq |n| + 1$ *then* $n_1 | n$ *and* $n_2 | n$.

Proof Suppose to the contrary (without loss of generality) $n_1 \not| n$. Then $n_1 n_2 = n + kq$ for some integer $k \neq 0$. Noting $0 < n < q$, so

$$|n_1| + |n_2| \geq |n_1 n_2| = |n + kq| \geq |q| - 1 > |n| + 1, \tag{17}$$

contradicting the condition $|n_1| + |n_2| \leq |n| + 1$. □

3.1 The Proposed Scheme

We are now ready to present our secret factorization distribution scheme. Again, let Alice be the prover (user) and Bob be the verifier, who, for instance, can be a key certification authority.

Protocol P Verifiable Encryption of Integer Factorization.
Task In this protocol Alice shall encrypt two prime numbers P and Q and discloses $N = PQ$ to Bob. Bob shall verify the correctness of the encryption under an agreed public key.
Data preparation by Alice Generates two primes P, Q, $abs(|P| - |Q|) < C$, and C is a pre-specified small constant; in practice, $C \leq 20$). Computes $V_1 = g^P (\mathrm{mod} p)$, $V_2 = g^Q (\mathrm{mod} p)$, and $N = PQ$. Encrypts P in A_1, B_1, and Q in A_2, B_2 as follows:

$$(A_1 = h^{K_1},\ B_1 = Y^{-K_1} P)\ (\mathrm{mod}\, q), \tag{18}$$

$$(A_2 = h^{K_2},\ B_2 = Y^{-K_2} Q)\ (\mathrm{mod}\, q) \tag{19}$$

where $K_1, K_2 \in_R Z_r$, and Y is the agreed public key in (12). She also prepares bit-commitment values. We will postpone the description of the bit-commitment procedure to Section 3.2.

The protocol steps are as follows.

1. Alice sends to Bob: A_1, B_1, V_1, A_2, B_2, V_2, $(K_1 + K_2)(\mathrm{mod}\, r)$ and N.
2. Bob verifies that

$$(h^{K_1+K_2} = A_1 A_2,\ \ N = Y^{K_1+K_2} B_1 B_2)\ \ (\mathrm{mod}\, q). \tag{20}$$

3. Alice shows to Bob evidence that N consists of only two distinct primes. We assume that Alice has constructed N to be a Blum integer [2]. Then there exists interactive or non-interactive protocols for showing such evidence [9, 13]. (We will discuss some special structure of N in Section 3.3.)
4. Alice proves to Bob that (A_1, B_1) encrypts $\log_g(V_1)$, and (A_2, B_2) encrypts $\log_g(V_1)$, and the encryptions are under the agreed public key Y. This uses the non-interactive prove technique described in Section 2.3.
5. Alice shows to Bob $|P|$ and $|Q|$ using a sub-protocol to be described in the next subsection. He verifies that

$$|P| + |Q| \leq |N| + 1 \tag{21}$$

and

$$abs(|P| - |Q|) < C. \tag{22}$$

6. Bob accepts the proof if every checking in the above passes, or else rejects. Terminate.

Upon successful termination of a run, Bob will certify N as Alice's public key and archive the A_1, B_1, V_1, A_2, B_2, V_2 for possible future recovery of P and Q.

We point out that if Protocol P and that for proof of N's two-prime-product structure are run in non-interaction. The cryptographicly strong hash functions used in the proofs act as publicly trustworthy challengers. In these cases, the verification procedures conducted by Bob are universally verifiable (by any third party). Collusion between Alice and Bob is computationally infeasible.

3.2 Proof of Bit-Lengths $|P|$ and $|Q|$

We now provide details for Alice to show $|P|$ and $|Q|$ which has been missing from Step 5 of Protocol P. Describing the case of $|P|$ suffices.

Let $m = |P| - 1$ and

$$P = a_0 2^0 + a_1 2^1 + \cdots + a_m 2^m \quad \text{for } a_i \in \{0, 1\} \text{ and } i = 0, 1, \cdots, m \tag{23}$$

be the binary presentation of P. Alice chooses $u_0, u_1, \cdots, u_m \in_R Z_q$. She computes

$$u = u_0 2^0 + u_1 2^1 + \cdots + u_m 2^m \mod q \tag{24}$$

and

$$E_i = E(a_i, u_i) = g^{a_1} f^{u_i} \bmod p, \quad \text{for } i = 0, 1, \cdots, m. \tag{25}$$

The computations in (25) follow a bit-commitment scheme [18], where f is the fixed element in G chosen as in Section 2.1. The prover commits herself to a_i (called *committal*) which is hidden under u_i. It can be proved that (i) the commitment $E(a_i, u_i)$ reveals no useful information about the committal a_i, and (ii) the prover cannot commit to two $a_i' \neq a_i$ using the same E_i unless she can find $\log_g(f)$. We shall see more about this in a moment.

Alice shall send E_i and u to Bob. The verification step for Bob is as follows. He checks whether

$$V_1 f^u \stackrel{?}{=} \prod_{i=0}^{m} E_i^{2^i} \mod p. \tag{26}$$

Then for each E_i $(i = 0, 1, \cdots, m)$, Alice and Bob shall run a sub-protocol (Bit) to prove one-bit-length of the committal a_i. Protocol Bit shows either $E_i = g f^{u_i}$ or $E_i = f^{u_i}$ without revealing which one is the case. This protocol is obtained by applying the transformation of [3] on the Schnorr identification protocol [23] on the instance $(E_i, E_i/g)$, with a 1-out-of-2 threshold scheme to show that the prover knows the discrete logarithm of either inputs without revealing which is the case. It has many useful applications (see e.g., [1, 4]).

Protocol Bit
Common input: $E, f, g \in G$
Prover's input: $z \in Z_q$
To prove either $E = f^z$ or $E = gf^z$

Prover (Alice)		Verifier (Bob)
$E = f^z$ | $E = gf^z$		
$w, r_1, c_1 \in_R Z_q$ | $w, r_2, c_2 \in_R Z_q$		
$a = f^w$ | $a = f^{r_2} E^{-c_2}$		
$b = f^{r_1}(E/g)^{-c_1}$ | $b = f^w$		
(above computing in mod p)	$\xrightarrow{a, b}$	
		$c \in_R Z_q$
	$\xleftarrow{c}$	
$E = f^z$ | $E = gf^z$		
$c_2 = c - c_1$ | $c_1 = c - c_2$		
$r_2 = w + zc_2$ | $r_1 = w + zc_1$		
(above computing in mod q)	$\xrightarrow{r_1, r_2, c_1, c_2}$	
		$c \stackrel{?}{=} c_1 + c_2 (\mathrm{mod} q)$
		$f^{r_1} \stackrel{?}{=} b(E/g)^{c_1} (\mathrm{mod} p)$
		$f^{r_2} \stackrel{?}{=} aE^{c_2} (\mathrm{mod} p)$

Bob will accept $|P| = m + 1$ if the result of running Protocol Bit is accepted for each E_i $(i = 0, 1, \cdots, m)$. We reason about the security of the bit-commitment scheme below.

Firstly, if P and u are indeed the numbers that Alice has constructed in (23) and (24), then indeed

$$\prod_{i=0}^{m} E_i^{2^i} = g^{a_0 2^0 + a_1 2^1 + \cdots + a_m 2^m} f^{u_0 2^0 + u_1 2^1 + \cdots + u_m 2^m} = V_1 f^u \ (\mathrm{mod}\, p). \tag{27}$$

On the other hand from (25) and (27) we note that

$$V_1 f^u (\mathrm{mod} p) = E(P, u). \tag{28}$$

So if Alice is able to find a $P' \neq P (\mathrm{mod}\, q)$ such that $E(P, u) = E(P', u')$, then it has to be $u' \neq u (\mathrm{mod}\, q)$ and

$$\log_g(f) = \frac{P - P'}{u' - u} \mod q. \tag{29}$$

This means that Alice knows $\log_g(f)$, contradicting the assumption that this value is known to nobody. Therefore, P in (23) and u in (24) give the only way for Alice to demonstrate $V_1 f^u = E(P, u) \ (\mathrm{mod}\, p)$.

We recall that the bit-commitment $E_i = E(a_i, u_i)$ protects the confidentiality of the bit information a_i. Therefore Bob gets no knowledge about P from E_i (for $i = 0, 1, \cdots, m$), but $|P|$.

Finally we point out that following the technique of [23], the three-move protocol (Bit) can easily be turned into non-interactive by using a hash function to generate the challenge as in the standard method for parallelizing an interactive proof. A non-interactive proof is universally verifiable.

3.3 Key Recovery

The factorization of N can be recovered if the shareholders decrypt (A_1, B_1) and (A_2, B_2) via distributed computing. The protocol that proves two-prime-product structure of N is up to showing that $N = R^t S^u$ for some odd positive integers t, u and distinct primes R, S [9]. So it remains to complete to factor N down to the primes R and S in order to compute the trapdoor information. Below we show that it is easy to complete the factorization.

Without loss of generality, we can assume that Peggy has constructed P (and Q) such that each of them consists of both R and S, maybe powers of them. Then $gcd(P, Q)$ will return either (i) R^i, or (ii) S^j, or (iii) $R^k S^\ell$ for some positive integers i, j, k and ℓ. For the first two cases, binary search for R from R^i in the case (i) for each fixed i takes at most $\log_2(P)$ steps and there are only at most $\log_2(P)$ many $i > 1$ to search. In reality, i should be sufficiently small or else R will be too small for the resulting modulus N to be secure. Analogously for the case (ii) we can efficiently search S out of S^j. In the sequel, whenever we meet a single prime or a power of it, we will regard the job has been done.

Now we consider the case (iii): $gcd(P, Q) = R^k S^\ell$ for $k, \ell > 1$. Let $a := P/gcd(P, Q)$ and $b := Q/gcd(P, Q)$. Then a and b are co-prime, and they cannot both be 1 because $P \neq Q$ or else N is a square and this is impossible since s and t are odd. Thus we will have, by symmetry (without loss of generality), further two sub-cases. The first one is that a is a single prime or a power of it, and we have done. The second sub-case is that a is a composite and $b = 1$. Then obviously we will have $P = aQ$. Noting the bit-length condition (22), $abs(|P| - |Q|) < C$, we will have

$$|a| = (|Q| + |a|) - |Q| \leq abs(|P| - |Q|) + 1 < C + 1.$$

So a will have a length less than $C + 1$ bits. Here C is a small constant, typically between 1 and 20. It is trivially easy to factor a into primes.

Thus, no matter how Peggy has constructed $N = PQ$, key recovery will always return two distinct primes.

3.4 Performance

We now provide analyses on the performance of the scheme.

The scheme consists of three components which are computation intensive. They are the three proof of knowledge steps regarding the factorization of $N = P * Q$: (i) verifiable encryption of P and Q; (ii) proof of the bit lengths of P and Q; and (iii) proof of the Blum integer structure of N.

The complexity of (i) can be measured as follows. Let the hash function $\mathcal{H}$ output 160 bits. When verifiably encrypting a number, for each bit of the hash

function output, both prover and verifier should compute 3 exponentiations. Thus for P and Q, they shall each compute $2 * 3 * 160 = 960$ exponentiations. The size of data transmitted is bounded by $2 * 160 * |p|$ bits.

The complexity of (ii) is proportional to the bit length of N. From Protocol Bit we can see that for each bit of N, both prover and verifier shall compute 4 exponentiations. Also 7 numbers of length $|q|$ will be exchanged between them. Therefore, the total number of exponentiations to be computed is $4 * |N|$ by each of the parties, and they exchange $7 * |N| * |q|$ bits. Similar to the case of (i), the proof can be parallelized using a hash function.

Finally, the complexity of (iii). Protocols for proof of the Blum integer structure of N ([9, 13]) require a prover to send roughly 100 integers modulo N to a verifier for a simple procedural checking, and are very efficient. These proofs are also nun-interactive, and universally verifiable.

From the analyses we conclude that the scheme may be considered to have an acceptable performance for sharing of long-term keys, while is not very practical for session keys.

4 Conclusion

We have presented a fair public-key cryptosystem that uses two-party protocols to distribute the factorization of an integer to multi-shareholders without their participation in receiving and verifying the shares. To the author's knowledge the proposed scheme is the first factorization-based fair cryptosystem that uses un-participating shareholders. Because the share distribution can be universally verifiable, the correctness of secret sharing is guaranteed. Further research in improving the performance will be desirable.

Acknowledgments Discussions with Kenneth Paterson and Dipankar Gupta of HP Labs., Bristol improved a few technical details. Colin Boyd of Queens University of Technology, Brisbane provided useful comments on a later draft of the paper.

References

1. M. Bellare and S. Goldwasser. Verifiable partial key escrow. In Proceedings of *4th ACM Conference on Computer and Communications Security*. Zurich, April 1997.
2. M. Blum. Coin flipping by telephone: a protocol for solving impossible problems. In Proceedings of *24th IEEE Computer Conference (CompCon)*, pages 133–137. 1982.
3. R. Cramer, I. Damgård and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology — Proceedings of CRYPTO'94 (LNCS 839)*, pages 174–187. Springer-Verlag, 1994.
4. R. Cramer, R. Gennaro and B. Schoenmakers. A secure and optimally efficient multi-authority election scheme. In *Advances in Cryptology — Proceedings of EUROCRYPT'97 (LNCS 1233)*, pages 103–118. Springer-Verlag, 1997.

5. D. Denning and D. Branstad. A taxonomy for key escrow encryption systems. *Communications of the ACM.* 39,3 March 1996, pages 34–40.
6. Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *Advances in Cryptology — Proceedings of CRYPTO'89 (LNCS 435)*, pages 307–315. Springer-Verlag, 1990.
7. T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, July 1985.
8. A. Fiat and A. Shamir. How to prove yourself: Practical solution to identification and signature problems. In *Advances in Cryptology — Proceedings of CRYPTO'86 (LNCS 263)*, pages 186–194. Springer-Verlag, 1987.
9. J. van de Graaf and R. Peralta. A simple and secure way to show the validity of your public key. In *Advances in Cryptology — Proceedings of CRYPTO'87 (LNCS 293)*, pages 128–134. Springer-Verlag, 1988.
10. J.A. Gordon. Strong primes are easy to find. In *Advances in Cryptology — Proceedings of EUROCRYPT'84 (LNCS 209)*, pages 216–223. Springer-Verlag, 1985.
11. J. Kilian and T. Leighton. Fair cryptosystems, revisited. A Rigorous approach to key-escrow. In *Advances in Cryptology — Proceedings of CRYPTO'95 (LNCS 963)*, pages 208–221. Springer-Verlag, 1995.
12. A.K. Lenstra, P. Winkler and Y. Yacobi. A key escrow system with warrant bounds. In *Advances in Cryptology — Proceedings of CRYPTO'95 (LNCS 963)*, pages 197–207. Springer-Verlag, 1995.
13. S. Micali. Fair public key cryptosystems. In *Advances in Cryptology — Proceedings of CRYPTO'92 (LNCS 740)*, pages 113–138. Springer-Verlag, 1993.
14. T. Okamoto. An efficient divisible electronic cash scheme. In *Advances in Cryptology — Proceedings of CRYPTO'91 (LNCS 963)*, pages 438–451. Springer-Verlag, 1995.
15. T. Okamoto. Threshold key-recovery system for RSA. In Proceedings of *1997 Security Protocols Workshop.* Paris. April, 1997.
16. T. Pedersen. Distributed provers with applications to undeniable signatures. In *Advances in Cryptology — Proceedings of EUROCRYPT'91 (LNCS 547)*, pages 221–242. Springer-Verlag, 1991.
17. T. Pedersen. A threshold cryptosystem without a trusted party. In *Advances in Cryptology — Proceedings of EUROCRYPT'91 (LNCS 547)*, pages 522–526. Springer-Verlag, 1991.
18. T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology — Proceedings of CRYPTO'91 (LNCS 576)*, pages 129–140. Springer-Verlag, 1992.
19. L.C. Guillou and J.-J. Quisquater. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In *Advances in Cryptology — Proceedings of EUROCRYPT'88 (LNCS 330)*, pages 123–128. Springer-Verlag, 1988.
20. R.L. Rivest, A. Shamir and L.M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* v.21, n.2, 1978, pages 120–126.
21. M.O. Rabin. Digital signatures and public-key functions as intractable as factorization. MIT Laboratory for Computer Science, Technical Report, MIT/LCS/TR-212. 1979.
22. A. Shamir. How to share a secret. *Communications of the ACM* 22, 1979, pages 612–613.

23. C.P. Schnorr. Efficient signature generation for smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.
24. M. Stadler. Publicly Verifiable Secret Sharing. In *Advances in Cryptology — Proceedings of EUROCRYPT'96 (LNCS 1070)*, pages 190–199. Springer-Verlag, 1996.