Efficient Publicly Verifiable Secret Sharing Schemes with Fast or Delayed Recovery

Fabrice Boudot and Jacques Traoré

France Telecom - CNET
42, rue des Coutures
BP 6243, F-14066 Caen Cedex 4, France
fabrice.boudot@cnet.francetelecom.fr
jacques.traore@cnet.francetelecom.fr

Abstract. A publicly verifiable secret sharing scheme is a secret sharing scheme in which everyone, not only the shareholders, can verify that the secret shares are correctly distributed. We present new such schemes and use them to share discrete logarithms and integer factorizations. The shareholders will be able to recover their shares quickly (fast recovery) or after a predetermined amount of computations (delayed recovery) to prevent the recovery of all the secrets by un-trustworthy shareholders (e.g. if these schemes are used for escrowing secret keys). The main contribution of this paper is that all the schemes we present need much less computations and communicated bits than previous ones [BGo, FOk, Mao, Sta, YYu]. By the way, we introduce in this paper several tools which are of independent interest: a proof of equality of two discrete logarithms modulo two different numbers, an efficient proof of equality of a discrete logarithm and a third root, and an efficient proof of knowledge of the factorization of any composite number n, where it is not necessary to prove previously that n is the product of two prime factors.

Keywords: Publicly Verifiable Secret Sharing, Key Escrow Systems, Zero Knowledge Proofs.

1 Introduction

In a secret sharing scheme, a *dealer* wants to split his secret s between l *shareholders*. Each shareholder receives through a secure channel his share s_i. If the dealer is honest, the collaboration of at least k shareholders allows to recover the secret s, while the collaboration of only $k-1$ shareholders does not reveal any extra-information about s. The first secret sharing scheme has been presented by [Sha].

Unfortunately, if the dealer is dishonest, the collaboration of the shareholders fails to retrieve the secret s. To overcome this problem, Chor et al. proposed a verifiable secret sharing in [CGMA], where the shareholders can verify the validity of their shares (i.e. that at least k of them can recover the secret s).

Another property, called *public verifiability*, was introduced by Stadler in [Sta]. In a publicly verifiable secret sharing (PVSS) scheme, everyone, not only the shareholders, can verify the validity of the shares.

This is an informal description of a PVSS scheme: Each shareholder SH_i has a public-key encryption function $\mathcal{E}_i$. The dealer wants to share a secret s, which is a discrete logarithm or the factorization of a large number. The dealer splits his secret into l shares $s_1, \ldots, s_l$, computes for all i $E_i = \mathcal{E}_i(s_i)$ and a *certificate of recoverability* $Cert$ which proves that the E_i are correctly computed. Then, he sends $(E_1, \ldots, E_l, Cert)$ to a verifier who checks the validity of the certificate of recoverability. When the shareholders want to recover the dealer's secret, the verifier sends E_i to each shareholder SH_i who can recover s_i. Finally, the collaboration of at least k shareholders allows to recover the dealer's secret s.

PVSS schemes can be used to build a key escrow system [Mic]. In such systems, the user is allowed to use cryptologic products by his government only if he discloses his secret key to a *trusted third party*. This user plays the role of the dealer who shares his secret key, the certification authority plays the role of the verifier and certifies the user's public key only if the certificate of recoverability is valid, and the escrow authorities play the role of the shareholders who recover the user's key on court order.

In some applications, it seems desirable that the shareholders need a certain amount of work to recover the secret key. For example, in a key escrow system, this avoids that an inquisitive government be able to recover a lot of keys in short time. We will call such a system a PVSS scheme with delayed recovery.

Our work: We present quite efficient PVSS schemes to share discrete logarithms and integer factorizations, the recovery of which can be made fast or delayed. In particular, these schemes can be used for sharing RSA keys.

Organization of the paper: In Section 2, we give some definitions and recall previous results about these topics. Then, in Section 3, we present the basic tools we will use in our schemes. We describe these schemes in Section 4 and compare their efficiency. Finally, we conclude in Section 5.

2 Background

The following definition is derived from [YYu].

Definition 1: A $(k,l)-$*Publicly Verifiable Secret Sharing* (PVSS) scheme is a triple (GEN, VER, REC) such that:

1. GEN is a poly-time probabilistic Turing machine, run by the dealer, that takes no input and generates $(K_P, K_S, (E_1, \ldots, E_l), Cert)$. K_P is the public key of the dealer, K_S his private key (or his secret), $(E_1, \ldots, E_l)$ the messages dedicated to the shareholders and $Cert$ the certificate of recoverability.
2. VER is a poly-time probabilistic Turing machine, run by the verifier, that takes $(K_P, (E_1, \ldots, E_l), Cert)$ as its input and returns a boolean value. With very high probability, VER returns true if and only if $(E_1, \ldots, E_l)$ can allow at least k shareholders to recover K_S.

3. REC is a set of probabilistic Turing machines $REC_1, \ldots, REC_l, END$. Each REC_i is a Turing machine with private input and belongs to the shareholder SH_i. If VER returns "true", then a) each REC_i, on input E_i, outputs a share s_i of K_S, b) on any set of k different elements from $\{s_1, \ldots, s_l\}$, END outputs K_S, c) the knowledge of less than k different elements from $\{s_1, \ldots, s_l\}$ gives no information about K_S.
4. It is intractable to recover K_S given $(K_P, (E_1, \ldots, E_l), Cert)$.

Definition 2:

1. A PVSS is a *PVSS with fast recovery* if the algorithms REC_i and END run rapidly.
2. A PVSS is a *PVSS with delayed recovery* if any set of k REC_i and/or END needs a non-trivial amount of work to output correct values.

In PVSS with delayed recovery, the shareholders are not able to recover a large number of private keys. This property was first discussed by Bellare and Goldwasser in [BGo]. In their schemes, the full secret is not shared, but only a part of it, and the shareholders are obliged to find the missing bits by exhaustive research to recover the full secret. Unfortunately, other schemes with delayed recovery may suffer from *early recovery*, i.e. one of the shareholders can recover by himself the missing bits. So, if the PVSS scheme is used for escrowing the users' private keys, and if one of the escrow authorities (shareholders) belongs to a political group with totalitarian projects, he can recover the missing bits of all users before seizing power and getting all the users' keys just after the coup, when the other escrow authorities are under his control. We propose another way to achieve delayed recovery: we will use an encryption scheme such that the decryption is very slow. Then each shareholder needs a lot of work to recover one share, and if a malicious shareholder recovers all its shares, this does not help him to recover faster the dealer's secret, as the time needed to recover a secret is the maximum of the times for computing each share (the shares are computed by each shareholder in parallel). Note that our definition of delayed recovery is less restrictive than [BGo]'s one: they claim that a PVSS does not suffer from early recovery only if the shareholders need a non-trivial amount of work to recover the secret *after* the cooperation of the shareholders (i.e. only if END runs slowly). In our definition, we also include the case when all the shares are recovered slowly.

Note that other works about delayed recovery have been done by [RSW].

Related results: An efficient PVSS scheme was presented by Stadler in [Sta], and allows to share a discrete logarithm. A similar result was presented by Young and Yung in [YYu]. Mao presented a PVSS scheme for factorization in [Mao]. All these results were improved by Fujisaki and Okamoto in [FOk]. A PVSS scheme with delayed recovery was presented by Bellare and Goldwasser in [BGo] for discrete logarithm. PVSS schemes with delayed recovery were presented by Fujisaki and Okamoto in [FOk], but the delayed recovery property is obtained only if the number of shareholders is about 48. We present in this paper two PVSS with fast recovery, one for discrete logarithm and the other for factorization. Our PVSS schemes are more efficient than the previous ones. We also present for both problems two PVSS with delayed recovery. They are more efficient than previous ones and allow any number of shareholders, including one only. Our schemes are at least 3 times more efficient in terms of bits communicated and 10 times more efficient in terms of computations needed than all the previous ones.

Notations: Throughout this paper, $\mathbb{Z}_n$ denotes the residue class ring modulo n, and $\mathbb{Z}_n^*$ denotes the multiplicative group of invertible elements in $\mathbb{Z}_n$. Let $\varphi(n)$ be the cardinality of $\mathbb{Z}_n^*$ (the Eulerian number of n) and $\lambda(n)$ be the maximal order of an element in $\mathbb{Z}_n^*$ (the Carmichael number of n). $|\cdot|$ denotes binary length, abs$(\cdot)$ denotes the absolute value, $a \parallel b$ is the concatenation of the strings a and b. We denote $\log_g(a)[\mathrm{mod}\ n]$ the discrete logarithm of a in base g modulo n which belongs to $\{-\mathrm{ord}(g)/2, \ldots, \mathrm{ord}(g)/2 - 1\}$. We note Proof$(\mathcal{R})$ a zero-knowledge proof of the assertion $\mathcal{R}$, and Proof$(x, \mathcal{R}(x))$ a zero-knowledge proof of knowledge of x such that $\mathcal{R}(x)$ is true.

3 Basic Tools

In this section, we present all the basic tools we will use in Section 4. Some of them are already known: a proof of knowledge of a discrete logarithm modulo a composite number by [Gir] in Section 3.1, that we extend to a proof that this logarithm is range-bounded, and a verifiable proof of secret sharing by [Fel] in Section 3.3.

Other ones are variants or extensions of existing schemes: We present a proof of equality of two discrete logarithms with respect to different moduli, in Section 3.2. This proof is related to the protocol due to [CPe] (but in their protocols the moduli are the same) and also to a protocol due to [Bao] (where the order of the bases are different, one known and the other unknown). We also present a proof of equality of a discrete logarithm and a third root related to [FOk] in Section 3.6. We will introduce a cryptosystem with arbitrarily slow decryption in Section 3.7. Such a cryptosystem is used by [MYa] for other purposes.

The remaining tools are (to our knowledge) new: we will introduce in Section 3.4 a new problem equivalent to the factorization problem and use it in Section 3.5 to prove that the knowledge of a particular discrete logarithm allows to factor a given integer. This is an original method for sharing the factorization of an integer. The main advantage of our method is, unlike [Mao] and [FOk] schemes, that we do not have to prove that this integer is the product of exactly two primes. Indeed, the protocols that can be used to prove that an integer is the product of exactly two primes [vGP, Mic] are relatively inefficient.

All the proofs described in this section are made non-interactive using the Fiat-Shamir [FSh] heuristic.

3.1 Proof of Knowledge of a Range-Bounded Discrete Logarithm Modulo a Composite Number

The following protocol allows Alice to prove that she knows a discrete logarithm and that this discrete logarithm is range-bounded. It is an extension (related e.g. to [CFT]) of a previous protocol by Girault [Gir], proven secure in the random oracle model by Poupard and Stern [PSt].

Let n be a composite number whose factorization is unknown by Alice, and $g \in \mathbb{Z}_n^*$ an element of maximal order. Let $h : \{0,1\}^* \longrightarrow \{0,\ldots,k-1\}$ be a hash function.

Let b, B, t and k be four integers such that bt/B and $1/k$ are negligible. Assume that Alice knows an integer $x \in \{0,\ldots,b\}$ such that $y = g^x \bmod n$.

The following protocol is a non-interactive zero-knowledge proof that she knows a discrete logarithm of y in base g modulo n and that this common discrete logarithm is in $\{-B,\ldots,B\}$ (in default of proving that it belongs to $\{0,\ldots,b\}$).

Protocol: $\mathrm{Proof1}(x, y = g^x \bmod n \wedge \mathrm{abs}(x) < B)$

1. Alice chooses $r \in \{0,\ldots,tkb-1\}$ and computes $W = g^r \bmod n$.
2. Alice computes $c = h(W)$.
3. Alice computes $D = r + cx$ (in $\mathbb{Z}$) and sends the proof (W,c,D) if $D \in \{cb,\ldots,tkb-1\}$, otherwise she starts the protocol again.
4. The verifier checks that $W = g^D y^{-c} \bmod n$ and that $D \in \{cb,\ldots,tkb-1\}$.

3.2 Proof of Equality of Two Discrete Logarithms

We design in this paragraph a protocol which allows Alice to prove that she knows a discrete logarithm common to two (or more) values and that this common discrete logarithm is range-bounded.

Let n be a composite number whose factorization is unknown by Alice, and $g \in \mathbb{Z}_n^*$ an element of maximal order. Let n' be an integer, prime or composite, and $g' \in \mathbb{Z}_{n'}^*$ an element of maximal order. We assume that n and n' are equal or relatively prime. Let $h : \{0,1\}^* \longrightarrow \{0,\ldots,k-1\}$ be a hash function.

Let b, B, t and k be four integers such that bt/B and $1/k$ are negligible. Assume that Alice knows an integer $x \in \{0,\ldots,b\}$ such that $y = g^x \bmod n$ and $y' = g'^x \bmod n'$.

The following protocol is a non-interactive zero-knowledge proof that she knows a discrete logarithm common to y in base g modulo n and to y' in base g' modulo n' and that this common discrete logarithm is in $\{-B,\ldots,B\}$ (in default of proving that it belongs to $\{0,\ldots,b\}$).

The first proof of equality of two discrete logarithms is due to [CEG], and has been improved by [CPe] and [Bao] (which prove the equality of two discrete logarithms modulo the same number).

Protocol: $\mathrm{Proof2}_b(x, y = g^x \bmod n \wedge y' = g'^x \bmod n' \wedge \mathrm{abs}(x) < B)$

1. Alice chooses $r \in_R \{1, \ldots, tkb - 1\}$ and computes $W = g^r \bmod n$ and $W' = g'^r \bmod n'$.
2. Alice computes $c = h(W, W')$.
3. Alice computes $D = r + cx$ (in $\mathbb{Z}$) and sends the proof (W, W', c, D) if $D \in \{cb, \ldots, tkb - 1\}$, otherwise she starts the protocol again.
4. The verifier checks that $D \in \{cb, \ldots, tkb - 1\}$, $W = g^D y^{-c} \bmod n$ and $W' = g'^D y'^{-c} \bmod n'$.

Lemma 1: The above protocol is a non-interactive zero-knowledge proof system, i.e.:

1. *Completeness*: If Alice behaves honestly and if $x \in \{0, \ldots, b\}$, then the protocol succeeds with probability greater than $1 - 1/t$.
2. *Soundness*: If Alice does not know any common discrete logarithm, then the protocol succeeds with probability less than $2/k$. Moreover, if Alice succeeds, then $\mathrm{abs}(x) < B$ with probability greater than $1 - bt/B$.
3. *Non-Interactive Zero-Knowledge*: There exists a simulator in the random oracle model which outputs conversations indistinguishable from the true ones. In the random oracle model, the simulator is allowed to simulate the random oracle, i.e. the hash function h [BRo].

How to use Proof2 when $n \neq n'$: If the verifier is convinced by Proof2 that there exists a common discrete logarithm of y and of y', then he knows that someone who is able to compute $\log_{g'}(y')[\bmod\ n']$ is able to compute $\log_g(y)[\bmod\ n]$. Assume that $|\lambda(n')| = |n| + \beta$, where $\beta \geq 91$. Let $b = n$ and $B = \lambda(n')/2$. For $|t| = 10$, the protocol proves that $\mathrm{abs}(x) > \lambda(n')/2$ with probability less than $2tn/\lambda(n')$, and so less than 2^{-80} which is negligible. Let $\tilde{x} = \log_{g'}(y')[\bmod\ n']$, Proof2 convinces the verifier that $\tilde{x} = x \bmod \lambda(n')$ and that $\mathrm{abs}(x) < \lambda(n')$. Hence, $\tilde{x} = x$ in $\mathbb{Z}$ and then someone who is able to compute $\tilde{x}$ is able to compute a discrete logarithm of y in base g modulo n.

In the rest of this paper, we will assume that $t = 2^{10}$. Then, if $b/B < 2^{-91}$, Alice can prove that $\mathrm{abs}(x) < B$ when she uses a secret $x \in \{0, \ldots, b\}$.

Note: Such a protocol can be used to prove the equality of more than two discrete logarithms.

Correctness of lemma 1:

Completeness: If Alice behaves honestly, the protocol succeeds if and only if $D \in \{cb, \ldots, tkb\ -1\}$, therefore if $r \in \{cb - cx, \ldots, tkb - 1 - cx\}$. As r is taken randomly and uniformly over $\{1, \ldots, tkb - 1\}$ and x belongs to $\{0, \ldots, b\}$, the probability that the protocol succeeds is greater than $(tkb - cb)/(tkb - 1) > (tk - c)/tk > 1 - 1/t$

Soundness: We first show that the discrete logarithms are equal with probability greater than $1 - 2/k$, using [PSt] lemma : if Alice knows $(g, y, c_1, c_2, D_1, D_2)$ such that $g^{D_1} y^{-c_1} \equiv g^{D_2} y^{-c_2} \pmod{n}$, then she knows a discrete logarithm of y in base g unless she can factor n.

Case 1: $n = n'$

Assume that Alice can, given W and W', compute correct proofs (W, W', c_1, D_1) and (W, W', c_2, D_2) with $c_1 \neq c_2$. Then, we have $W = g^{D_1} y^{-c_1} \bmod n =$

$g^{D_2}y^{-c_2} \bmod n$ and $W' = g'^{D_1}y'^{-c_1} \bmod n = g'^{D_2}y'^{-c_2} \bmod n$. Using [PSt] lemma, such equalities imply that Alice knows $x = \log_g(y) = \log_{g'}(y')$.
Case 2: n and n' are relatively prime.

Let ϕ be the Chinese Remainder Theorem isomorphism :

$$\begin{array}{rcl}\phi : \mathbb{Z}^*_{nn'} & \longrightarrow & \mathbb{Z}^*_n \times \mathbb{Z}^*_{n'} \\ x & \longmapsto & (x \bmod n, x \bmod n')\end{array}$$

We note $\bar{W} = \phi^{-1}(W, W')$, $\bar{g} = \phi^{-1}(g, g')$ and $\bar{y} = \phi^{-1}(y, y')$. If the proof succeeds, then $\bar{W} = \bar{g}^D \bar{y}^{-c} \bmod nn'$.

Assume that Alice can, given W and W', compute correct proofs (W, W', c_1, D_1) and (W, W', c_2, D_2). Then we have : $\bar{W} = \bar{g}^{D_1}\bar{y}^{-c_1} \bmod nn' = \bar{g}^{D_2}\bar{y}^{-c_2} \bmod nn'$. Using [PSt] lemma, this equality implies that Alice knows $x = \log_{\bar{g}}(\bar{y})[\bmod nn']$ unless she can factor n. So she knows the discrete logarithm $x = \log_g(y)[\bmod n] = \log_{g'}(y')[\bmod n']$.

To prove that $\text{abs}(x) < B$ with probability less than tb/B, it is sufficient to prove that for all u, $\Pr[success/\text{abs}(x) > tub] < 1/u$. Let $\bar{r} = \log_{\bar{g}}(\bar{W})[\bmod nn']$, then we have $D = \bar{r} + xc$. Alice succeeds only if $D \in \{cb, \ldots, tkb - 1\}$, i.e. c must be in the set $I = \{\lceil \bar{r}/x \rceil, \ldots, \lfloor (\bar{r} - tkb - 1)/x \rfloor\}$. If $\text{abs}(x) > tub$, her probability of success is $\Pr(success/\text{abs}(x) > tub) = (\sharp I)/k < (k/u)/k = 1/u$.
Non-Interactive Zero-Knowledge : The simulator picks random $c \in \{0,1\}^k$ and $D \in \{cb, \ldots, tkb - 1\}$ and sets $W = g^D y^{-c} \bmod n$, $W' = g'^D y'^{-c} \bmod n'$ and $h(W, W') = c$.

3.3 Verifiable Proof of Secret Sharing

In [Sha], Shamir presents a (k, l)-threshold secret sharing in which a secret s is split into l shares such that the secret can be computes if k shares are known, but no information about s can be computed if only $k - 1$ shares are known.

Let p and q be two large prime numbers such that $p = 2q + 1$, g be an element of order q in $\mathbb{Z}^*_p$ and $s \in \mathbb{Z}_q$ the secret a dealer wants to share between l shareholders. The following protocol shares s into l shares $s_1, \ldots, s_l$.

Protocol: $Share(s) = ((s_1, \ldots, s_l), (a_1, \ldots, a_{k-1}))$

1. For $j = 1, \ldots, k - 1$, pick random $a_j \in \mathbb{Z}_q$.
2. Set $P(X) = s + \sum_{j=1}^{k-1} a_j X^j \bmod q$.
3. For $i = 1, \ldots, l$ compute $s_i = P(i)$.

If k shares are known, then s can be recovered using Lagrange's interpolation.

Let $S = g^s \bmod p$, $S_i = g^{s_i} \bmod p$ for $i = 1, \ldots, l$ and $A_j = g^{a_j} \bmod p$ for $j = 1, \ldots, k - 1$. The following protocol due to [CPS] and [Fel] proves that $\log_g(S_i)$ are the shares.

Protocol : Proof3($Share(\log_g(S)) = ((\log_g(S_1), \ldots, \log_g(S_l)), (\log_g(A_1), \ldots, \log_g(A_k - 1)))$

1. The dealer sends $(S, S_1, \ldots, S_l, A_1, \ldots, A_k - 1)$ to the verifier.
2. The verifier checks that for all $i = 1, \ldots, l$, $S_i = S \cdot \prod_{j=1}^{k-1} (A_j)^{i^j} \bmod p$.

3.4 A Problem Equivalent to Factorization

Let n be a composite number, e be coprime with $\lambda(n)$ and d be such that $de \equiv 1 \bmod \lambda(n)$. We prove in this subsection the equivalence between the factorization and the knowledge of the common discrete logarithm of g_i in base g_i^e for $i = 1, 2$, where g_i is computed (but not freely chosen) by the dealer who knows the factorization of n. We will prove that this common discrete logarithm allows to factor n.

Lemma: Assume that a time-bounded dealer has chosen a composite number n and has computed $g_i = h(n \parallel i) \bmod n$ for $i = 1, 2$, where h is a hash function which outputs $|n|$-bit values. Let e be coprime with $\lambda(n)$. Assume that this dealer has disclosed to a party an integer α, a common discrete logarithm of g_i in base g_i^e for $i = 1, 2$. Then this party, on input α, can factor n. In other words, the dealer cannot compute in polynomial time a composite number n such that a common discrete logarithm of g_i in base g_i^e for $i = 1, 2$ does not reveal the prime factors of n.

Sketch of proof: First note that the dealer can compute such α if and only if e is coprime with $\lambda(n)$: d such that $de \equiv 1 \pmod{\lambda(n)}$ satisfies the required properties.

We first state this result (that we do not prove for lack of space): let n be a composite number and g be an element randomly selected over $\mathbb{Z}_n^*$. If β divides $\lambda(n)$, the probability that $\lambda(n)/\mathrm{ord}(g)$ is equal to β or a multiple of β is less than or equal to $1/\beta$.

Once the dealer has chosen the composite number n, he has no control on the values of g_1 and g_2 (due to the use of the hash function h). So, given β, his only chance to get (n, g_1, g_2) such that both $\lambda(n)/\mathrm{ord}(g_1)$ and $\lambda(n)/\mathrm{ord}(g_2)$ are equal to β or a multiple of β is to try different values of n until $g_i^{\lambda(n)/\beta} = 1$ for $i = 1, 2$, which happens after more than β^2 tests on average: as the probability that $\mathrm{ord}(g_i)$ is equal to β or a multiple of β is less than $1/\beta$ for $i = 1, 2$ and that these probabilities are independent, the probability that both $\mathrm{ord}(g_1)$ and $\mathrm{ord}(g_2)$ are equal to $\lambda(n)/\beta$ or a divisor of $\lambda(n)/\beta$ is less than $1/\beta^2$.

Remark : The cheating dealer can also take several values β_j ($j = 1..l$) which divide $\lambda(n)$ (e.g. all the divisors of $\lambda(n)$ less than $\lambda(n)/2^{40}$) and test for all β_j if $g_i^{\lambda(n)/\beta_j} = 1$ for $i = 1, 2$. Then, his probability of success is $\sum_j 1/\beta_j^2$ which is less than $l/(\min(\beta)^2)$, but the number of tests "$g_i^{\lambda(n)/\beta} = 1$" is multiplied by l. Thus, this strategy is worse than the previous one.

So, as the dealer is time-bounded, we assume that it is infeasible for the dealer to try more than β^2 different values of n (for example, $\beta > 2^{40}$).

Let $L = \mathrm{lcm}(\mathrm{ord}(g_1);\mathrm{ord}(g_2))$ and $M = \lambda(n)/L$. As the dealer is time-bounded, we can assume that he is unable to find n such that M is a multiple of β. As α is a discrete logarithm of g_i^e in base g_i for $i = 1, 2$, we have $g_i^{e\alpha} = g_i \bmod n$ and thus $e\alpha = 1 \bmod \mathrm{ord}(g_i)$ for $i = 1, 2$. Consequently, $e\alpha = 1 \bmod L$; so there exists $k \in \mathbb{Z}$ such that $e\alpha - 1 = kL$ (in $\mathbb{Z}$) and hence $M(e\alpha - 1) \equiv 0 \bmod \lambda(n)$.

To factor n, we first choose an element y in $\mathbb{Z}_n^*$, we compute M', a discrete logarithm of 1 in base $y^{e\alpha-1}$ (which is M or a divisor of M). This discrete logarithm is less than β, so it can be computed using $2\sqrt{\beta}$ modular multiplications using Shanks [Sch] algorithm. Then, $M'(e\alpha - 1) \equiv 0 \bmod \mathrm{ord}(y)$. So we get $V_1 = M'(e\alpha - 1)$, a multiple of $\mathrm{ord}(y)$.

Then, we choose another element z and we compute M'', the discrete logarithm of 1 in base z^{V_1}. So we get $V_2 = M'M''(e\alpha - 1)$, a multiple of $\mathrm{ord}(y)$ and of $\mathrm{ord}(z)$. Note that as y was randomly selected over $\mathbb{Z}_n^*$, its order α is such that $\lambda(n)/\alpha$ is very small. So M'', which divides $\lambda(n)/\alpha$, is very small too and can be computed by exhaustive search.

We repeat this process until we are convinced that there exists no element whose order does not divides V_j (i.e. even the elements of maximal order). This is the case after 80 rounds, as the probability that K elements do not generate the whole group $\mathbb{Z}_n^*$ is less than $\zeta(K) = \sum_{i=2..\infty} 1/i^K \approx 2^{-K}$ (according to the result stated at the beginning of this proof). Then, V_j is a multiple of $\lambda(n)$ and the party can factor n by Miller's lemma [Mil] (it is well known that given n and a multiple of $\lambda(n)$ we can factor n even if n has more than two prime factors).

Efficiency of the computation of a multiple of the order of y with respect to the computational effort of a cheating dealer: We recall that a cheating dealer can output a composite n such that $\mathrm{lcm}(\mathrm{ord}(g_1);\mathrm{ord}(g_2))$ is greater than β only after trying and testing β^2 moduli n, and that, in this case, we need $2\sqrt{\beta}$ modular multiplications using Shanks algorithm to be able to compute a multiple of the order of y. For $\beta = 2^{40}$, the cheating dealer needs 2^{80} tries (which is infeasible nowadays) and Shanks algorithm requires 2^{21} modular multiplications (which can be done in a few seconds). This fact will discourage any dealer from attempting to cheat. And, if the dealer does not cheat, M is very small (in 99% of the cases, M is less than 10) and can be found by exhaustive search.

3.5 Proof That a Discrete Logarithm Allows to Factor n

In this subsection, we give an efficient proof of the knowledge of a discrete logarithm and that this discrete logarithm allows to factor n. In this proof, it is unnecessary to prove that n is the product of only two prime factors, unlike [Mao] and [FOk] schemes. This avoids to use inefficient protocols to prove this fact, like [Mic] or [vGP]. Note that, for this reason, this proof is the more efficient proof of knowledge of the factorization of a composite number n.

Let p be a large prime number, and g an element of maximal order in $\mathbb{Z}_p^*$, N be a large composite number whose factorization is unknown by all the participants (the dealer, the verifier and the shareholders) which is generated by a trusted third party, and G an element in $\mathbb{Z}_N^*$ (randomly selected by the shareholders). Assume that Alice knows the factorization of a large composite number n. Let d be such that $de \equiv 1 \pmod{\lambda(n)}$, and $Y = g^d \bmod p$. If $n/\mathrm{ord}(g) < 2^{-91}$, the following protocol allows Alice to prove that $\log_g(Y)[\bmod p]$ allows to factor n.

Protocol: Proof5($d, Y = g^d \bmod p \wedge d$ allows to factor n)

1. Alice computes $g_1 = h(n \parallel 1) \bmod n$ and $g_2 = h(n \parallel 2) \bmod n$.

2. Alice sends $W = G^d \bmod N$.
3. Alice executes Proof2$_n(d, Y = g^d \bmod p \wedge g_1 = (g_1^e)^d \bmod n \wedge g_2 = (g_2^e)^d$ mod$n \wedge W = G^d \bmod N \wedge$ abs$(d) <$ord$(g)/2)$.

Correctness : First note that Proof2 (which is sound only if the factorization of one of the moduli is unknown by Alice) can be used because the factorization of N is unknown by Alice (this is the only utility of the number W). Let α be the common discrete logarithm of Y in base g modulo n, g_1 in base $g_1^e \bmod n$ and g_2 in base $g_2^e \bmod n$. Using the method described in Section 3.4., it is possible, knowing α, to factor n.

3.6 Proof of Equality of a Discrete Logarithm and a Third Root

Let p be a large prime number and g an element of maximal order in Z_p^*. Let n_1 be a large composite number such that its factorization is unknown by Alice. We assume that $p/n_1 < 2^{-100}$.

Given G and E, the following protocol proves that there exists x such that $G = g^x \bmod p$, $E = x^3 \bmod n_1$ and that abs$(x) < (n_1 - 1)/2$. This protocol is related to [FOk].

Let N be a large composite number, whose factorization is unknown by all the participants, such that $N < n_1$ and that $p/\lambda(N) < 2^{-91}$, and $\bar{g}$ an element in Z_N^*. The protocol runs as follows:

Protocol: Proof6$(x, G = g^x \bmod p \wedge E = x^3 \bmod n_1 \wedge$abs$(x) < (n_1 - 1)/2)$

1. Alice computes $\alpha = \frac{E - x^3}{n_1}$, $G_1 = \bar{g}^x \bmod N$, $G_2 = \bar{g}^{x^2} \bmod N$, $G_3 = \bar{g}^{x^3} \bmod N$ and $Z = \bar{g}^{\alpha n_1} \bmod N$.
2. Alice computes Proof2$_p(x, G = g^x \bmod p \wedge G_1 = \bar{g}^x \bmod N \wedge G_2 = G_1^x \bmod N \wedge G_3 = G_2^x \bmod N \wedge abs(x) < \lambda(N)/2)$ and Proof1$(\alpha, Z = (\bar{g}^{n_1})^\alpha \bmod N)$.
3. The verifier checks Proof2 and Proof1, computes $T = \bar{g}^E \bmod N$ and checks that $G_3 = T/Z \bmod N$.

Correctness: If all the proofs succeed, then Alice knows x such that $G = g^x \bmod p$, $G_1 = \bar{g}^x \bmod N$, $G_3 = \bar{g}^{x^3} \bmod N$ and such that $abs(x) < \lambda(N)$,and α such that $Z = \bar{g}^{\alpha n_1} \bmod N$. Hence, $\bar{g}^{x^3} = T/\bar{g}^{\alpha n_1} \bmod N$, and this implies $x^3 \equiv E - \alpha n_1 (\bmod \lambda(N))$. So $x^3 = E - \alpha n_1$ (if not, Alice would be able to factor N). It comes $E \equiv x^3 (\text{mod} n_1)$.

How to use Proof6: If the verifier is convinced by Proof6 that there exists x such that $G = g^x \bmod p$ and $E = x^3 \bmod n_1$ and abs$(x) < (n_1 - 1)/2$, then he knows that someone who is able to computes the third root of E is able to compute the discrete logarithm of G in base g modulo p.

Let $\tilde{x}$ be the third root of E, which is congruent to x modulo n_1. As it is proven that $abs(x) < \lambda(N) < n_1/2$, this third root (which theoretically belongs to $\{(-n_1 + 1)/2, \ldots, (n_1 - 1)/2\}$) is equal (in $\mathbb{Z}$) to the value x used by Alice in this protocol. So this value is the discrete logarithm of G_1 in base $\bar{g}$ modulo N (which belongs to $\{-\lambda(N)/2, \ldots, \lambda(N)/2\}$). Then, this value is also a discrete logarithm of G in base g modulo p, and it is easy to compute the discrete logarithm of G in base g modulo p, which is $x \bmod \mathrm{ord}(g)$.

Secrecy of x: Although x has only $|p| = 1024$ bits and n has 1500 bits, to compute x from $E = x^3 \bmod n_1$ is infeasible.

Thus, the third root of E is a discrete logarithm of G in base g modulo p.

3.7 A Cryptosystem with Arbitrarily Slow Decryption

We characterize composite numbers n for which solving the discrete logarithm modulo n is hard, but becomes easier when its factorization is known. Such numbers are used by Maurer and Yacobi [MYa] to provide an identity-based cryptosystem.

Let $n = pq$ such that $p-1 = 2p_1 \dots p_r$ and $q-1 = 2q_1 \dots q_{r'}$. We denote $m = \min(\max(p_i); \max(q_i))$ and $M = \max(\max(p_i); \max(q_i))$. Let g be an element of maximal order in $\mathbb{Z}_n^*$. In practice, $|n| = 1500$ and $|m| \simeq |M| \simeq 70$.

Solving a discrete logarithm modulo n is hard: This problem is harder than factoring n. As $p-1$ and $q-1$ have relatively small prime factors, some factorization algorithms become more efficient, like Pollard's algorithm [Pol]. Such algorithms run in time $O(m)$, so our choice of $|m|$ makes the factorization infeasible.

When the factorization is known, solving the problem is feasible: The Pohlig-Hellman's algorithm [PHe] consists of computing the discrete logarithm modulo each p_i and each q_i using Shanks [Shn] algorithm. Shanks algorithm is used $r + r' \simeq |n|/|M|$ times and needs $2\sqrt{M}$ modular exponentiations, each of them costs $3|M|/2$ modular multiplications. So, it finds the discrete logarithm in about $3|n|\sqrt{M}$. If we choose $|M| = 70$ and $|n| = 1500$, the algorithm needs 2^{47} modular multiplications.

Then, Alice can encrypt x setting $E = g^x \bmod N$ and Bob, who knows the factorization of n, can compute $x = \log_g(E)[\bmod n]$ using Pohlig-Hellman algorithm.

In the rest of this paper, we will call such number a Maurer-Yacobi (MY) number.

4 Our Schemes

4.1 How to Share a Discrete Logarithm with Fast Recovery

Let $p = 2q + 1$ be a prime number, g be an element of order q in $\mathbb{Z}_p^*$, N be a composite number whose factorization is unknown by Alice, G be an element of maximal order in $\mathbb{Z}_N^*$ and, for $i = 1, \dots, l$, n_i be a composite number whose factorization is only known by the shareholder SH_i. Typically, $|p| = 1024$, and $|n_i| = 1500$. Let s be Alice's secret and $S = g^s \bmod p$ be a public number.

To compute the shares and the certificate:

1. *Share:* Alice runs $Share(s) = ((s_1, \dots, s_l), (a_1, \dots, a_{k-1}))$, computes $S_i = g^{s_i} \bmod p$ for $i = 1, \dots, l$ and $A_j = g^{a_j} \bmod p$ for $j = 1, \dots, k-1$. Then

she executes Proof3($Share(\log_g(S)) = ((\log_g(S_1), \ldots, \log_g(S_l)), (\log_g(A_1), \ldots, \log_g(A_{k-1})))$.

2. *Encrypt each share:* For $i = 1, \ldots, l$, Alice computes $E_i = s_i^3 \bmod n_i$ and executes Proof6($s_i, S_i = g^{s_i} \bmod p \wedge E_i = s_i^3 \bmod n_i \wedge \text{abs}(s_i) < (n_i - 1)/2$).

To check the validity of the certificate:
The verifier checks that all proofs succeed.
To recover the secret:
At least k shareholders decrypt E_i and recover s_i, then they can recover s using Lagrange's interpolation.

4.2 How to Share a Factorization with Fast Recovery

Let $P = 2Q + 1$ be a prime number, g be an element of order Q in $\mathbb{Z}_P^*$, and, for $i = 1, \ldots, l$, n_i be a composite number whose factorization is only known by the shareholder SH_i. Let p, q be Alice's secret and $n = pq$ be a public number. Let d such that $de \equiv 1 \pmod{\lambda(n)}$. Typically, $|n| = 1024$, $|P| = 1400$ and $|n_i| = 1500$.

To compute the shares and the certificate:

1. *Transform the secret into a discrete logarithm :* Alice sets $S = g^d \bmod P$ and executes Proof5($d, S = g^d \bmod P \wedge d$ allows to factor n).
2. *Share:* Alice runs $Share(d) = ((s_1, \ldots, s_l), (a_1, \ldots, a_{k-1}))$, computes $S_i = g^{s_i} \bmod P$ for $i = 1, \ldots, l$ and $A_j = g^{a_j} \bmod P$ for $j = 1, \ldots, k-1$. Then she executes Proof3($Share(\log_g(S)) = ((\log_g(S_1), \ldots, \log_g(S_l)), (\log_g(A_1), \ldots, \log_g(A_{k-1})))$.
3. *Encrypt each share:* For $i = 1, \ldots, l$, Alice computes $E_i = s_i^3 \bmod n_i$ and executes Proof6($s_i, S_i = g^{s_i} \bmod P \wedge E_i = s_i^3 \bmod n_i \wedge \text{abs}(s_i) < (n_i - 1)/2$).

To check the validity of the certificate:
The verifier checks that all proofs succeed.
To recover the secret:
At least k shareholders decrypt E_i and recover s_i, then they can recover, using Lagrange's interpolation, a number which allows to factor n.

4.3 How to Share a Discrete Logarithm with Delayed Recovery

Let $p = 2q + 1$ be a prime number, g be an element of order q in $\mathbb{Z}_p^*$, for $i = 1, \ldots, l$, n_i be a MY-number whose factorization is only known by the shareholder SH_i, and g_i be an element of maximal order in $\mathbb{Z}_{n_i}^*$. Typically, $|p| = 1024$ and $|n_i| = 1500$. Let s be Alice's secret and $S = g^s \bmod p$ be a public number.

To compute the shares and the certificate:

1. *Share:* Alice runs $Share(s) = ((s_1, \ldots, s_l), (a_1, \ldots, a_{k-1}))$, computes $S_i = g^{s_i} \bmod p$ for $i = 1, \ldots, l$ and $A_j = g^{a_j} \bmod p$ for $j = 1, \ldots, k-1$. Then she executes Proof3($Share(\log_g(S)) = ((\log_g(S_1), \ldots, \log_g(S_l)), (\log_g(A_1), \ldots, \log_g(A_{k-1})))$.

2. *Encrypt each share:* For $i = 1, \ldots, l$, Alice computes $E_i = g_i^{s_i} \bmod n_i$ and executes $\text{Proof2}_p(s_i, S_i = g^{s_i} \bmod p \wedge E_i = g_i^{s_i} \bmod n_i \wedge \text{abs}\ (s_i) < \lambda(n_i)/2)$.

To check the validity of the certificate:
The verifier checks that all proofs succeed.
To recover the secret:
At least k shareholders decrypt E_i and recover s_i using Pohlig-Hellman algorithm, then they can recover s using Lagrange's interpolation.

4.4 How to Share a Factorization with Delayed Recovery

Let $P = 2Q + 1$ be a prime number, g be an element of order Q in $\mathbb{Z}_P^*$, for $i = 1, \ldots, l$, n_i be a MY-number whose factorization is only known by the shareholder SH_i, and g_i be an element of maximal order in $\mathbb{Z}_{n_i}^*$. Let p, q be Alice's secret and $n = pq$ be a public number. Let d such that $de \equiv 1 \pmod{\lambda(n)}$. Typically, $|n| = 1024$, $|P| = 1400$ and $|N| = |n_i| = 1500$.
To compute the shares and the certificate:

1. *Transform the secret into a discrete logarithm:* Alice sets $S = g^d \bmod P$ and executes $\text{Proof5}(d, S = g^d \bmod P \wedge d \text{ allows to factor } n)$.
2. *Share:* Alice runs $Share(d) = ((s_1, \ldots, s_l), (a_1, \ldots, a_{k-1}))$, computes $S_i = g^{s_i} \bmod P$ for $i = 1, \ldots, l$ and $A_j = g^{a_j} \bmod P$ for $j = 1, \ldots, k-1$. Then she executes $\text{Proof3}(Share(\log_g(S)) = ((\log_g(S_1), \ldots, \log_g(S_l)), (\log_g(A_1), \ldots, \log_g(A_{k-1})))$.
3. *Encrypt each share:* For $i = 1, \ldots, l$, Alice computes $E_i = g_i^{s_i} \bmod n_i$ and executes $\text{Proof2}_P(s_i, S_i = g^{s_i} \bmod P \wedge E_i = g_i^{s_i} \bmod n_i \wedge \text{abs}\ (s_i) < \lambda(n_i)/2)$.

To check the validity of the certificate:
The verifier checks that all proofs succeed.
To recover the secret:
At least k shareholders decrypt E_i and recover s_i using Pohlig-Hellman algorithm, then they can recover, using Lagrange's interpolation, a number which allows to factor n.

4.5 Efficiency of Our Schemes

We evaluate the efficiency of each scheme and compare it to the efficiency of [FOk]'s schemes. The values are given for $k = 5$ and $l = 3$ (5 shareholders, such that at least 3 can recover the secret) as an example, but the savings are of the same order for any values of k and l.

This first table gives the number of 1024-bit multiplications the dealer must perform to compute its certificate.

Property of the scheme	Efficiency of [FOk] scheme (mult.)	Efficiency of our scheme (mult.)	% comp. saved
DL, fast recovery	144.10^3	16.10^3	88.9%
Fact, fast recovery	185.10^3	19.10^3	89.7%
DL, delayed recovery	144.10^3	11.10^3	92.4%
Fact, delayed recovery	185.10^3	14.10^3	92.4%

This second table gives the length of the message the dealer must send to the verifier.

Property of the scheme	Efficiency of [FOk] scheme (kBytes)	Efficiency of our scheme (kBytes)	% bits saved
DL, fast recovery	30.8	11.7	62.0%
Fact, fast recovery	40.8	12.3	69.9%
DL, delayed recovery	30.8	3.81	87.6%
Fact, delayed recovery	40.8	4.43	89.1%

DL = sharing of a discrete logarithm
RSA = sharing of the factorization

5 Conclusion

We have presented publicly verifiable secret sharing schemes for both discrete logarithm and integer factorization problems, and with fast or delayed recovery. Our schemes allow to save about 90% of computations with respect to the previous ones, and more than 62% of bits communicated. Moreover, there are no restricted conditions (such that 48 shareholders for schemes with delayed recovery) in our schemes, unlike [FOk] schemes.

Acknowledgements

We would like to thank Marc Girault for helpful discussions and comments.

References

[BGo] M. Bellare and S. Goldwasser, Verifiable Partial Key Escrow, *Proceedings of the Fourth Annual Conference on Computer and Communications Security, pp. 78-91*, 1997.

[Bao] F. Bao, An Efficient Verifiable Encryption Scheme for Encryption of Discrete Logarithms, *to be published in the proceedings of CARDIS'98*, 1998.

[BRo] M. Bellare and P. Rogaway, Random Oracles are Practical: a Paradigm for Designing Efficient Protocols, *Proceedings of the First Annual Conference and Communications Security, pp. 62-73*, 1993.

[CEG] D. Chaum, J.-H. Evertse and J. van de Graaf, An Improved Protocol for Demonstrating Possesion of Discrete Logarithm and Some Generalizations, *Proceedings of EUROCRYPT'87, LNCS 304, pp. 127-141*, 1988

[CFT] A. Chan, Y. Frankel and Y. Tsiounis, Easy Come - Easy Go Divisible Cash, *Proceedings of EUROCRYPT'98, LNCS 1403, pp. 561-575*, 1998.

[CGMA] B. Chor, S. Goldwasser, S. Micali and B. Awerbuch, Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults, *Proceedings of FOCS, pp. 383-395*, 1985.

[CPe] D. Chaum and T.P. Pedersen, Wallet Databases with Observers, *Proceedings of CRYPTO'92, LNCS 740, pp. 89-105*, 1992.

[CPS] C. Charnes, J. Pieprzyk and R. Safavi-Naini, Conditionally Secure Secret Sharing Scheme with Disenrolment Capability, *Second ACM Conference on Computer and Communication Security, pp. 89-95*, 1994.

[Fel] P. Feldman, A Practical Scheme for Non-Interactive Verifiable Secret Sharing, *Proceedings of the 28th IEEE Symposium on FOCS, pp. 427-437*, 1987.

[FOk] E. Fujisaki and T. Okamoto, A Practical and Provably Secure Scheme for Publicly Verifiable Secret Sharing and Its Applications, *Proceedings of EUROCRYPT'98, LNCS 1403, pp. 32-46*, 1998.

[FSh] A. Fiat and A. Shamir, How to Prove Yourself: Practical Solutions to Identification and Signature Problems, *Proceedings of CRYPTO'86, LNCS 263, pp. 186-194*, 1986.

[Gir] M. Girault, Self-Certified Public Keys, *Proceedings of EUROCRYPT'91, LNCS 547, pp. 490-497*, 1991.

[vGP] J. van de Graaf and R. Peralta, A Simple and Secure Way to Show the Validity of Your Public Key, *Proceedings of CRYPTO'87, LNCS 293, pp. 128-134*, 1987.

[Mao] W. Mao, Guaranteed Correct Sharing of Integer Factorization with Off-line Shareholders, *Proceedings of Public Key Cryptography 98, pp. 27-42*, 1998.

[Mic] S. Micali, Fair Public-Key Cryptosystems, *Proceedings of CRYPTO'92, LNCS 740, pp. 113-138*, 1992.

[Mil] G. Miller, Riemann's hypothesis and Tests for Primality, *Journal of Computer and System Sciences (13), pp. 300-317*, 1976.

[MYa] U. Maurer and Y.Yacobi, Non-Interactive Public-Key Cryptography, *Proceedings of EUROCRYPT'91, LNCS 547, pp. 498-507*, 1991.

[PHe] S.C. Pohlig and M. Hellman, An Improved Algorithm for Computing Logarithms over $GF(p)$ and its Cryptographic Significance, *Proceedings of IEEE Transactions on Information Theory, Vol. IT-24, pp. 106-110*, 1978.

[Pol] J.M. Pollard, Theorems on Factorization and Primality Testing, *Proceedings of Cambridge Philos. Society, vol. 76, pp. 521-528*, 1974.

[PSt] G. Poupard and J. Stern, Security Analysis of a Practical "on the fly" Authentication and Signature Generation, *Proceedings of EUROCRYPT'98, LNCS 1403, pp. 422-436*, 1998.

[RSW] R. Rivest, A. Shamir and D. Wagner, Time-Lock Puzzles and Time-Release Crypto, *this paper is available at* `http://theory.lcs.mit.edu/~rivest/publications.html`, to appear.

[Sha] A. Shamir, How to Share a Secret, *CACM, vol. 22, No. 11, pp. 612-613*, 1979.

[Shn] D. Shanks, Five Number-Theoric Algorithms, *Proceedings of the 2nd Manitoba conference on numerical mathematics, pp. 51-70*, 1972.

[Sta] M. Stadler, Publicly Verifiable Secret Sharing, *Proceedings of EUROCRYPT'96, LNCS 1070, pp. 190-199*, 1996.

[YYu] A. Young and M. Yung, Auto-Recoverable Auto-Certifiable Cryptosystems, *Proceedings of EUROCRYPT'98, LNCS 1403, pp. 17-31*, 1998.