Verifiable Partial Escrow of Integer Factors

Wenbo Mao
Trusted E-Services Laboratory
HP Laboratories Bristol
HPL-2000-155
December 12th, 2000*

partial key escrow, RSA, zero-knowledge protocols

We construct an efficient interactive protocol for realizing verifiable partial escrow of the factors of an integer n with time-delayed and threshold key recovery features. The computational cost of the new scheme amounts to $10k \log_2 P$ multiplications of numbers of size of P, where P is a protocol parameter which permits n of size up to $(\log_2 P)$-4 to be dealt with and k is a security parameter which controls the error probability for correct key escrow under $1/2^k$. The new scheme realizes a practical method for fine tuning the time complexity for factoring an integer, where the complexity tuning has no respect to the size of integer.

* Internal Accession Date Only
Approved for External Publication

Verifiable Partial Escrow of Integer Factors

Wenbo Mao

Hewlett-Packard Laboratories
Filton Road, Stoke Gifford
Bristol BS34 8QZ
United Kingdom
wm@hplb.hpl.hp.com

November 16, 2000

Abstract

We construct an efficient interactive protocol for realizing verifiable partial escrow of the factors of an integer n with time-delayed and threshold key recovery features. The computational cost of the new scheme amounts to $10k \log_2 P$ multiplications of numbers of size of P, where P is a protocol parameter which permits n of size up to $(\log_2 P) - 4$ to be dealt with and k is a security parameter which controls the error probability for correct key escrow under $1/2^k$. The new scheme realizes a practical method for fine tuning the time complexity for factoring an integer, where the complexity tuning has no respect to the size of the integer.

Keywords Partial key escrow, RSA, Zero-knowledge protocols.

1 Introduction

In this article we construct an efficient interactive protocol for realizing verifiable partial escrow of the factors of an integer which features time-delayed and threshold key recovery.

The idea of partial key escrow was initiated by Shamir [25]. On recognition that an abusive government in control of an ordinary key escrow scheme may conduct privacy intrusion against mass individual citizens, Shamir suggested partial key escrow as a countermeasure. In partial key escrow a user's secret key is split into two parts, denoted

by subkey-1 and subkey-2 the two parts. Subkey-1 will be escrowed in an ordinary key escrow fashion (usually with a group of distributed escrow agents), while subkey-2 will be rendered missing. At the time of key recovery, only subkey-1 can be reconstructed algorithmically by the escrow agents using their key recovery data. Recovery of the whole key is then the task of finding subkey-2 via brute-force search which needs a length of time determined by the size of subkey-2. Thus, at the time of key escrow, the key escrow agents must have made sure that the missing key part (subkey-2) has an agreed size which will permit them later to recover it at an affordable cost. On the other hand the affordable cost should be non-trivial so that it will cause a prohibitive difficulty, even for well-resourced government agencies, to recover a mass number of keys in a short period of time. Nevertheless, partial key escrow will preserve the property of ordinary key escrow in speedy recovery of a small number of keys by resourceful agencies. Shamir's partial key escrow scheme was proposed for partial escrowing of a DES key.

Partial key escrow has also been investigated by Micali [19] and Bellare and Goldwasser [1, 2] with emphasis on public-key cryptosystems. These authors introduced and addressed an important issue: verifiability. The verifiability in partial key escrow goes beyond Micali's earlier idea of "fair public key cryptosystems" [18] and extends from merely fair sharing of secret to a guaranteed correctness of the size of a missing key part, subkey-2. It is necessary for a partial key escrow scheme to have this sense of verifiability since an incorrect size of subkey-2 will render impossible the finding of it via key search.

Bellare and Goldwasser observed a further important issue in partial key escrow: an attacking scenario which they termed "early key recovery attack" [1]. A partial key escrow scheme will be susceptible to such an attack if the scheme provides information that can be used for searching for subkey-2 before recovery of subkey-1. If such information is available then the key search can take place off-line and on a massive scale with many or all users in the system targeted. This essentially nullifies the effect of partial key escrow since the missing key part is no longer really missing and the whole secret key of a user can be made readily available straight after the recovery of subkey-1. (The verifiable partial key escrow scheme suggested by Micali [19] is susceptible to early key recovery attack [1].) Thus, during the time of key escrow, a partial key escrow scheme must not disclose any information that can be used for determination of subkey-2 before recovery of subkey-1. To meet this requirement is often a non-trivial problem and apparently the difficulty lies in the need of meeting the other requirement: making available the information of the size of subkey-2 at the same time rendering the latter missing.

1.1 Partial Escrow of Integer Factors

Keys in different cryptosystems take different structures. As a result there seems no general way to construct a partial key escrow scheme by following the basic idea of

splitting a secret key into two parts with one of them escrowed and the other made missing. Constructions of partial key escrow schemes vary for different cryptosystems and some of them turn out to be more difficult to realize than others.

A particularly difficult case of cryptosystems is those based on the problem of integer factorization. The private key of such a cryptosystem is essentially the prime factors of a given integer. It has been understood that partial key escrow for integer factoring based cryptosystems should not be based on directly splitting a prime factor of the integer because it is well-known that partial information of a prime factor can produce efficient algorithms to factor the integer (for instance, for $n = pq$ with p and q primes of roughly equal size, knowledge of up to half the bits of p will suffice to factor n in polynomial time in the size of n, see Coppersmith [8]). In general, for an integer of size secure against modern factorization methods and for a part of its prime factor rendered missing which has a size small enough to search back at an affordable cost, it is likely that the available part of the factor will be large enough to allow the whole prime to be found in an efficient manner. In other words, rendering a part of a prime factor missing will not result in a sound time complexity problem upon which to base a partial key escrow scheme.

The only known previous work on partial key escrow for integer factoring based cryptosystems is due to Bellare and Goldwasser [2]. Their approach is called "encapsulated key escrow" (EKE) and uses a primitive called an EKE time capsule. An EKE time capsule encrypts a secret in a timed-release manner, which means that its decryption procedure has a specified time complexity. In their EKE scheme for RSA, a key owner, who has generated an RSA modulus n, will create a list of EKE time capsules. Each capsule encrypts, respectively, one of two different square roots, mod n, of a quadratic residue element of the multiplicative group modulo n. The correctness of the encryption is checked in a cut-and-choose manner. Here the correctness means two things: first, the decryption of an EKE time capsule will reveal two square roots, mod n, of the same quadratic residue element and the two roots have different Jacobi symbols; secondly, the encryption key to form each time capsule conforms to an agreed size and thereby the decryption procedure will require time defined by the size of the key. It is well known that two such square roots will suffice to factor n with a simple calculation. During the cut-and-choose checking phase, a group of agents who are responsible for escrowing the key material will ask the key owner to open a random subset of the time capsules (they let the key owner disclose the respective encryption keys) for correctness checking. They will also perform secret sharing schemes to share each of the unchecked capsules as a secret. Later in the key recovery time, these agents can collectively reconstruct one of the shared capsule and thereby enable a time-delayed procedure to open it. Upon opening of the capsule, two square roots of the same quadratic residue element becomes available and n is factored. In this way, time-delayed factorization of n, or partial key escrow for RSA, is realized.

The need of checking the correct formation for a subset of time capsules is due to a high probability of error inhere in the cut-and-choose technique (in association with this is the key owner's interest in introducing errors deliberately). Obviously, enlarging the number of checkings (i.e., the size of the subset of the checked capsules) will reduce the probability of error and the reduction reaches the highest rate if the number of the checked capsules equals that of the unchecked ones. Let $k/2$ be the number of the checked and the unchecked capsules. After $k/2$ cut-and-choose checkings, the probability of error is (by Stirling's approximation, see, e.g., page 46 of [14])

$$\frac{1}{\binom{k}{k/2}} \approx 2^{-k}\sqrt{\frac{\pi k}{2}}. \tag{1}$$

For this value to be acceptably small (so that the key owner's successful cheating becomes unlikely), $k > 40$ is necessary. Consequently, in the EKE scheme, the secret sharing agents will have to perform more than 20 instances of secret sharing for the unchecked time capsules. Secret sharing, especially with a verifiable threshold recovery feature, is rather costly. High cost in achieving a verifiable threshold secret sharing is a major disadvantage of the EKE scheme for RSA.

1.2 Our Work

We propose a verifiable partial key escrow scheme for integer factoring based cryptosystems, which features (t, m)-threshold key recovery. (Here (t, m)-threshold key recovery means that any $t(< m)$ escrow agents can enable a key recovery procedure.) The new scheme is based on an observation that the time complexity (the time complexity is one of the cryptanalysis results of Mao and Lim reported in [16]) for factoring a composite integer n can be fine tuned by tuning the size of a factor of $\phi(n)$ (the Euler phi function). That factor will be in place of subkey-1, which has a proven size and will be shared in a verifiable threshold secret sharing scheme by a group of escrow agents.

Our scheme uses efficient zero-knowledge proof protocols for checking the correctness in key splitting. The computational cost for the protocols is measured by $10k \log_2 P$ multiplications of numbers of size of P, where P is a protocol parameter which permits n of size up to $(\log_2 P) - 4$ to be dealt with; here k is a security parameter which controls the probability of error under 2^{-k}. In the verifiable secret sharing part, there is only one piece of secret that is needed to share by the key escrow agents.

In comparison with the EKE scheme for RSA, our scheme forms a major cost reduction in the verifiable secret sharing part.[1] Based on similar probabilities of error:

[1] We only compare the secret sharing costs of the two schemes. Due to the multi-party nature of the EKE scheme's cut-and-choose protocol, we have not been able to provide a precise measurement on its computational cost, though it is easy to see a heavy communication overhead of that protocol.

2^{-k} against the value estimated in (1), the EKE scheme for RSA needs to share $k/2$ secrets while our scheme only needs to share one. This forms a big difference since verifiable threshold secret sharing is rather costly both in communication overhead and in computation.

In the remainder of this article: Section 2 describes an observation on a fine tunable time complexity for factorization; Section 3 constructs the proposed scheme; its security and performance are analyzed in Section 4 and Section 5, respectively; finally, Section 6 concludes the work.

2 Fine Tunable Time Complexity for Factorization

Let $n = pq$ for p and q being distinct primes. Then

$$n + 1 = (p - 1)(q - 1) + (p + q). \tag{2}$$

Let r be a factor of $\phi(n) = (p - 1)(q - 1)$. Then

$$p + q \equiv n + 1 \,(\mathrm{mod}\, r). \tag{3}$$

For $r \leq p + q$, congruence (3) implies

$$p + q = (n + 1 \mod r) + kr, \tag{4}$$

for an unknown k. If $p + q$ is known then factoring n becomes a simple job (see Steps 5 and 6 in Figure 1 for an algorithm). Equation (4) shows that finding the quantity $p + q$ with r known is equivalent to finding the unknown k and hence the difficulty of factoring n cannot be harder than that of finding k. Below we investigate the time complexity for finding k given the condition that p and q are of equal magnitude (this is compatible to a desired key structure for integer factoring based cryptosystems).

Combining (2) and (4), we have

$$n + 1 = (p - 1)(q - 1) + (n + 1 \mod r) + kr. \tag{5}$$

Let u be an element randomly picked from $(Z/nZ)^*$. Raising u to both sides of (5), writing $w = u^r \mod n$, and noticing $u^{(p-1)(q-1)} \equiv 1 \,(\mathrm{mod}\, n)$, we have

$$u^{n+1-(n+1 \bmod r)} \equiv w^k \,(\mathrm{mod}\, n). \tag{6}$$

If the order of u exceeds $p+q$ then k in (6) will be exactly the same as that in (4). This is because $kr \leq p + q < order(u)$ and so

$$kr \mod order(u) = kr,$$

$Time\_Tuned\_Factoring(r, n)$

1. pick $u < n$ at random;
2. $v := u^{[n+1-(n+1 \bmod r)]} (\bmod n)$;
3. $w := u^r (\bmod n)$;
4. extract the discrete logarithm of v to the base w modulo n (denote by k the returned discrete logarithm);
5. $m := (n + 1 \bmod r) + kr$;
6. solve quadratic equation (the two roots, denote by p, q, satisfy $n = pq$)

$$x^2 - mx + n = 0.$$

Figure 1: An algorithm to factor $n = pq$ with time tuned by a factor of $\phi(n)$

i.e., k will not be reduced as a result of transforming (4) to (6). Below we provide the probability for a uniform element $u \in (Z/nZ)^*$ to have order less than $p+q$. Let $p > q$. Then $2(p-1) > p-1+q-1$. In $(Z/nZ)^*$ there can be at most $2(p-1)$ elements of orders less than $p+q$ since these orders must divide $2(p-1)$ or $2(q-1)$. So for a uniform element $u \in (Z/nZ)^*$,

$$Pr[order(u) < p+q] \leq 2(p-1)/\phi(n) = 2/(q-1).$$

With p and q of equal magnitude this probability is negligible.

With r, and hence w (see step 3 in Algorithm $Time\_Tuned\_Factoring(r, n)$), known, to find k from (6) can be via to compute the discrete logarithm of $w^k (\bmod n)$ to the base w. For k of a relatively small size there exists a number of algorithms which can extract k in $O(\sqrt{k})$ multiplications (modulo n). Shank's baby-step giant-step algorithm (see e.g., [7], Section 5.4.1) provides a deterministic method with space requirement also measured by $O(\sqrt{k})$. Pollard's kangaroo algorithm [23] provides a probabilistic method and requires a trivial amount of space. Note that Pollard's rho method (e.g., page 106 of [17]) is not applicable here because it needs the order of w which is unknown in this case.

Figure 1 provides an algorithm which factors $n = pq$ using r, a known factor of $\phi(n)$.

Lemma 1 Let $n = pq$, p, q be distinct primes of equal magnitude, and r be a factor of $\phi(n)$. The time complexity of the algorithm $Time\_Tuned\_Factoring(r, n)$, measured in the number of multiplications, is bounded by $O(n^{1/4}/r^{1/2})$.

Proof For p and q of equal magnitude, $p+q \approx n^{1/2}$. From (4) $k \approx (p+q)/r \approx n^{1/2}/r$. The main computations in $Time\_Tuned\_Factoring(r, n)$ is in Step 4 for extracting k which requires $O(\sqrt{k})$ multiplications of integers modulo n. □

Remarks

1. Although Lemma 1 merely states an upper bound, we conjecture that the bound is also the lowest known to date with the following reasons. With the use of a sizable r, the difficulty for factoring n is reduced substantially from any methods that do not make use of r. The square-root reduction from finding k in (4) to extracting discrete logarithm k in (6) forms a further big step of complexity reduction. Any method for factoring n that does not exploit these two big steps of reductions yet still has a lower time complexity clearly forms a breakthrough in the integer factorization problem.

2. Van Oorschot and Wiener suggested a parallelized kangaroo algorithm [21] for extracting discrete logarithms. Using m processors that algorithm can shorten the computing time to $1/m$ of that of Pollard's original sequential algorithm (i.e., a linear speedup). However, our complexity bound $O(n^{1/4}/r^{1/2})$ shows that decreasing of the magnitude of r will cause increasing of the computing time in a sub-exponentiation rate; the effect of a linear speedup in any large scale can be suppressed easily. Moreover, we should also notice that the space requirement of Van Oorschot and Wiener's parallelized algorithm remains at the level of $O(n^{1/4}/r^{1/2})$ which is still prohibitively high. We emphasize that it is a high cost, rather than a un-by-passable time length, that is the necessary feature of a partial key escrow scheme.

3. A number of previous integer factoring based cryptosystems make use of a disclosed sizable factor of $\phi(n)$ (e.g., [10, 11, 13]). Each of these systems includes moduli settings that allow the factorization of the moduli in a feasible time. For instance, a modulus setting in [10] satisfies $n^{1/4}/r^{1/2} \approx 2^{35}$.

If r is unavailable, then equation (4), hence congruence (6), will not be usable. Factoring n is believed to be hard.

We will make use of the following two facts gained in this section as the underlying principles of a verifiable partial key escrow scheme for integer factoring based cryptosystems:

- there is a big difference in time complexity between factoring n with a sizeable factor of $\phi(n)$, and doing it without, and
- the time complexity for factoring n is fine tunable by the factor of $\phi(n)$.

Our realization will consist of sharing the factor r among a group of escrow agents with verifiability on its size and a proof of correct threshold secret sharing of it. Time delayed factorization of n will become possible after an agreed subset of the agents have recovered r.

3 The Proposed Scheme

Let user Alice construct her public key $n = pq$ such that p and q are prime, are of the same magnitude and that $(p-1)(q-1)$ has a factor r which makes $n^{1/4}/r^{1/2}$ sufficiently large yet still feasible to search by well resourced agencies. An instance of the magnitude of $n^{1/4}/r^{1/2}$ is 2^{40}.

Alice shall then prove in zero-knowledge the following things: (i) n is the product of two primes p and q of the same size; (ii) a number r is a factor of $(p-1)(q-1)$; (iii) prove the size of $n^{1/4}/r^{1/2}$ as agreed; (iv) verifiable secret sharing of r with a set of trustees (called shareholders). We notice that because the proofs in (i), (ii) and (iii) can be made into publicly verifiable (we will further clarify this in the end of Section 3.2 and in Section 4.1), therefore it suffices to use a single verifier (Bob) to verify Alice's proofs. Secret sharing in (iv) will be in threshold access structure, which allows a subgroup of shareholders to recover r as long as they satisfy the access structure.

After these have been done, we know from the previous section that once r is recovered by co-operative shareholders, n can be factored using $Time\_Tuned\_Factoring(r, n)$ in $O(n^{1/4}/r^{1/2})$ multiplications. On the other hand, if r is not available, the time complexity for factoring n will be measured purely by the size of n, and when n is sufficiently large, the factoring problem is infeasible.

In this section we will construct zero-knowledge proof protocols for the proof of the required structure of n, and for verifiable secret sharing of r. These constructions will use various previous results as building blocks, which can be found, respectively, in the work of Chaum and Pedersen [6] (for the proof of the discrete logarithm equality), Damgård [9] (for showing of integer sizes), Pedersen [22] (for verifiable threshold secret sharing), and Mao [15] (for the proof of correct integer arithmetic).

These protocols will make use of a public cyclic group with the following construction. Let P be a large prime such that $Q|P-1$ be also prime. Let $f \in (Z/PZ)^*$ be a fixed element of order Q. The public group is the group generated by f with multiplication as the group operation. We assume that it is computationally infeasible to compute discrete logarithms to the base f. Once setup, the numbers f and P and Q will be announced for use by the system wide entities.

3.1 Proof of Correct Integer Arithmetic

We shall apply $y \equiv f^x \pmod P$ as the one-way function needed to prove the correct integer arithmetic. For simplicity, in the sequel we will omit the operation mod P whenever the omission will not cause confusion.

For integers a and b, Alice can use the above one-way function to prove $c = ab$ without revealing a, b and c. She shall commit to the values a, b and c by sending to Bob (verifier) their one-way images $(A, B, C) = (f^a, f^b, f^{ab})$ and prove to him that the pre-image of C is the product of those of A and B.

Note that for the one-way function f^x used, the proved multiplication relationship in the exponents is in terms of modulo $ord(f)$ (here $ord(f) = Q$, the order of the element f), and in general this relationship does not demonstrate that $\log_f(C) = \log_f(A)\log_f(B)$ is also the case in the integer space. Nevertheless, if Alice can show the sizes of the respective discrete logarithms then the following lemma guarantees the correct multiplication (in the sequel we denote by $|a|$ the size of the number a in the binary representation).

Lemma 2 *Let* $ab = c \pmod Q$ *and* $|c| + 2 < |Q|$. *If* $|a| + |b| \leq |c| + 1$ *then* $ab = c$.

proof The inequality $ab \neq c$ implies $ab = c + \ell Q$ for some integer $\ell \neq 0$. Since $0 < c < Q$,

$$|a| + |b| \geq |ab| = |c + \ell Q| \geq |Q| - 1 > |c| + 1,$$

contradicting the condition $|a| + |b| \leq |c| + 1$. □

Thus $y = f^x$ does form a suitable one-way function to be used for the proof of the correct multiplication of the integers in its pre-image space provided the sizes of the pre-images are shown. Given f^x, there exists efficient protocols to show $|x|$ without revealing x (e.g., [9], we will specify such a protocol in the next subsection). Below we specify a protocol (predicate) $Product(A, B, C)$ to achieve a proof of the correct multiplication of the integers as the discrete logarithms of the input quantities. The predicate will return 1 (Bob accepts) if $\log_f(C) = \log_f(A)\log_f(B)$, or return 0 (Bob rejects) if otherwise.

Protocol $Product(A, B, C)$

(Abandon a run and return 0 if Bob finds any error in any checking step, otherwise return 1 upon termination.)

Alice sends to Bob: $|\log_f(A)|$, $|\log_f(B)|$, $|\log_f(C)|$, and demonstrates:

1. $\log_f(A) \equiv \log_B(C) \pmod Q$ and $\log_f(B) \equiv \log_A(C) \pmod Q$;
2. $|\log_f(A)| + |\log_f(B)| \leq |\log_f(C)| + 1 < |Q| - 1$.

In *Product*, showing the equalities in step 1 can use the protocol of Chaum and Pedersen [6], and showing the size information in step 2 can use a protocol *Bit-Size* to be specified in the next subsection. Note that only the bit sizes regarding the first two input values need to be shown because, having shown the equations in step 1, the size regarding the third value is always the summation of those regarding the former two.

3.2 Proof of Integer Size

The basic technique is due to Damgård [9]. We specify a simplified version based on the discrete logarithm problem (in Section 4.2 we shall see that our scheme is secure with respect to the Decision Diffie-Hellman problem in the group mod P).

Let I be the interval $[\xi, \eta] (= \{x | \xi \leq x \leq \eta\})$, $\theta = \eta - \xi$, and $I \pm \theta = [\xi - \theta, \eta + \theta]$. In Protocol *Bit-Size* specified below, Alice can convince Bob that the discrete logarithm of the input value to the agreed base lies in the interval $I \pm \theta$.

Protocol *Bit-Size*(f^s)

Execute the following k times:

1. Alice picks $0 < t_1 < \theta$ at uniformly random, and sets $t_2 := t_1 - \theta$; she sends to Bob the unordered pair $C_1 := f^{t_1}$, $C_2 := f^{t_2}$;

2. Bob selects challenge $c = 0$ or $c = 1$ at uniformly random, and sends c to Alice;

3. Alice sets
$$\begin{array}{llll} u_1 = t_1, & u_2 = t_2 & \text{for } c = 0, \\ u_1 = t_1 + s, & u_2 = t_2 + s & \text{for } c = 1, \end{array}$$
and sends u_1, u_2 to Bob;

4. Bob checks (with $i = 1, 2$)
$$\begin{array}{lll} -\theta < u_i < \theta, & C_i = f^{u_i} & \text{for } c = 0; \\ \xi - \theta \leq u_i \leq \eta + \theta, & C_i f^s = f^{u_i} & \text{for } c = 1; \end{array}$$

Let, for instance, $I = [2^{\ell-1},\ 2^{\ell}]$. Then $\theta = 2^{\ell-1}$, and *Bit-Size* will prove that the discrete logarithm of the input value does not exceed $\ell + 1$ bits. The probability for this to hold is at least $1 - 1/2^k$.

Using a secure one-way hash function to generate challenge bits, the proof becomes publicly verifiable. That is why we can trust Bob to be honest in correctly verifying Alice's proofs. (Using the same method, the proof of the discrete logarithm equality in *Product* can also be made publicly verifiable.)

3.3 Proof of the Structure of n

Let $n = pq$ be Alice's public key for an integer factoring based cryptosystem. We require Alice to set the primes p and q with the following structure

$$p = 2p's + 1, \quad q = 2q't + 1, \tag{7}$$

Here p', q', s, t are odd numbers, any two of them are relatively prime. Let

$$r = 4st$$

Then we have $r \mid \phi(n)$.

Let L be such that performing $2^{L/2}$ multiplications forms a non-trivial burden for even well resourced agencies ($L = 80$ is a typical setting). Alice should set the sizes for these quantities to satisfy

$$|p| = |q| \tag{8}$$

and

$$|p| - |r| = L. \tag{9}$$

As a recommended procedure for modulus setup, Alice should first choose the numbers p', q', s, t at random with the required sizes, parity and relative-prime relationship. She then samples if p and q in (7) are prime. The procedure repeats until p and q are found to be prime. For p', q', s, t being odd, both p and q will be congruent to 3 modulo 4, rendering n a Blum integer [4]. It is advisable that p' and q' be chosen as primes. Then for n of a secure size (at least of 512 bits), p' and q' will be sufficiently large which results in p and q as the so-called strong primes. This follows a desirable moduli setting for integer factoring based cryptosystems.

Once the above values are fixed, Alice shall publish

$$U = f^p, \quad V = f^q, \quad W = f^{\frac{(p-1)(q-1)}{r}}, \quad R = f^r.$$

Using these published values, Alice and Bob can run the following protocol steps:

1. $Product(U, V, f^n)$;
2. $Product(W, R, \frac{f^{n+1}}{UV})$.

Step 1 proves $n = \log_f(U)\log_f(V)$ and step 2 proves the fact that $\log_f(R)$ divides $(\log_f(U)-1)(\log_f(V)-1)$. Note that $\frac{f^{n+1}}{UV}$ is used as the third input to $Product$ in step 2 because its discrete logarithm is $n + 1 - (p + q) = (p - 1)(q - 1)$. During the proofs that use $Product$, Alice has also demonstrated the bit size values $|\log_f(U)|$, $|\log_f(V)|$ and $|\log_f(R)|$. Bob should check that (review (8))

$$|\log_f(U)| = |\log_f(V)|, \tag{10}$$

and that (review (9))

$$|\log_f(U)| - |\log_f(R)| = L. \tag{11}$$

Alice should finally prove that n consists of two prime factors only. This can be achieved by applying the protocol of Van de Graaf and Peralta [12] for the proof of Blum integers (we have constructed n to be a Blum integer), and the protocol of Boyar et al [5] for the proof of square-free integers.

From the size check in (10) Bob knows that the two prime factors of n are of the same size. Thus, $\log_f(U) \approx n^{1/2}$. Then the size check in (11) implies

$$[\log_f(U)/\log_f(R)]^{1/2} \approx 2^{L/2},$$

that is

$$n^{1/4}/r^{1/2} \approx 2^{L/2}. \tag{12}$$

Thus, once $r = \log_f(R)$ becomes available, to factor n using $Time\_Tuned\_Factoring(r, n)$ (in Figure 1) will indeed have a cost measured by $2^{L/2}$.

Finally we should note that for $n > 2^{512}$ (the least secure size in this date) and $L = 80$, (12) implies that the size of r is at the level of

$$|n|/2 - 80 > 256 - 80 = 176.$$

So r has an adequately large size against finding it from $R = f^r$ by extraction of a small discrete logarithm. For a larger n, (12) shows that r should further be increased accordingly.

3.4 Verifiable Threshold Secret Sharing of $r = \log_f(R)$

For self-containment, we include Pedersen's threshold verifiable secret sharing scheme [22] for sharing the secret r among a multi number of shareholders with threshold recoverability.

Let the system use m shareholders, Using Shamir's t ($< m$) out of m threshold secret sharing method ([24]), Alice can interpolate a t-degree polynomial $r(x)$

$$r(x) = \sum_{i=0}^{t-1} c_i x^i \mod Q,$$

where the coefficients $c_1, c_2, \cdots, c_{t-1}$ are randomly chosen from $(Z/QZ)^*$, and $c_0 = r$. The polynomial satisfies $r(0) = r$. She shall send the secret shares $r(i)$ $(i = 1, 2, \cdots, m)$ to each of the m shareholders, respectively (via secret channels), and publishes

$$f^{r(i)} \mod P, \quad \text{for } i = 0, 1, \cdots, m,$$

and

$$f^{c_j} \mod P \text{ for } j = 0, 1, \cdots, t-1.$$

Each shareholder can verify

$$f^{r(i)} \equiv \prod_{j=0}^{t-1} (f^{c_j})^{i^j} \pmod{P} \text{ for } i = 1, 2 \cdots, m.$$

Assume that at least t of the m shareholders are honest by performing correct checking. Then the polynomial $r(x)$ is now secretly shared among them. When key recovery is needed, they can use their secret shares to interpolate $r(x)$ and recover $r = r(0)$.

We should point out that this sub-protocol is not a necessary component in the proposed scheme. There exists other schemes to prove a correct threshold encryption of a discrete logarithm (e.g., [15] achieves verifiable secret sharing with a public verifiability; that is, secret sharing can be done without the presence of the m shareholders and without assuming t of them to be honest). We have chosen Pedersen's scheme for simplicity in presentation.

4 Security Analysis

In security analysis we are mainly concerned with two issues: correctness and privacy. In correctness we are concerned whether Alice is able to successfully cheat Bob to accept her false proofs, or whether they can collude to create false proofs, with the aim that the recovery of the prime factors of her modulus will have a much higher cost than that which is agreed in the prescribed key recovery procedure. In privacy we are concerned whether Bob can gain any useful knowledge, as a result of verifying Alice's proof, which leads to discovery of Alice's private key without going through the prescribed key recovery procedure.

4.1 Correctness

The correctness of the scheme lies in that of the protocols that our scheme applies to achieve a proof of a discrete logarithm equality, a proof of the size of an integer, a proof of the two prime product structure, and the verifiable secret sharing.

Firstly, the probability for a successful cheating in the proof of the discrete logarithm equality (protocol of Chaum and Pedersen [6]) is $1/Q$, where Q is the order of f. Since $Q > n$, this probability is adequately small (can be omitted in comparison with $1/2^k$). The probability for a successful cheating in the proof of the integer size (protocol of Damgård [9]) is $1/2^k$ where k is the number of challenging bits sent (either by Bob, or from the output of a secure one-way hash function).

Next, the correctness of the protocol for the proof of the two-prime product structure is also well established [5, 12], with error probability $1/2^k$ for k iterations of message verification. Here, k can be the same as that used in the size proof protocol.

We note that in the proofs of the correct structure of n (i.e., $n = pq$, the primality of p, q, the condition $r|(p-1)(q-1)$, and the required sizes of $n^{1/4}/r^{1/2}$), the verification job does not involve handling any secret information. Therefore it can be carried out by anybody and can be repeated if necessary. The standard way for making these protocols publicly verifiable is to use a secure one-way hash function to generate random challenge bits sent to Alice. The input to the hash function should include n and other public values such as U, V, W and R. The public verifiability means that Bob cannot collude with Alice in creating invalid proofs without being caught.

Finally, Pedersen's protocol [22] for the verifiable threshold secret sharing of r has a two-sided error. The error probability for Alice's successful cheating is $1/Q$. The other side of the error is determined by the number of dishonest shareholders. The number r can be correctly recovered if no fewer than t out of the m shareholders are honest (i.e., they follow the protocol and securely store the data for secret recovery). Usually the threshold value t is set to $\lfloor \frac{m}{2} \rfloor + 1$ to achieve a good tradeoff between reliability and fairness.

4.2 Privacy

Due to the scope and the space limits of this article, we shall only provide an informal discussion on the privacy quality of our scheme.

In the proof of the correct partial escrow of the prime factors of n, Alice has made the following quantities public

$$(U, V, W, R) = (f^p,\ f^q,\ f^{\frac{(p-1)(q-1)}{r}},\ f^r) \,(\mathrm{mod}\, P),$$

where f is an element in the multiplicative group modulo P and P is a large prime.

Evidently, were the above quantities not available, the prime factors of n are protected by the factorization problem. On the other hand, were n not available, given the disclosed quantities (U, V, W, R) to find (p, q, r) one faces the discrete logarithm problem. Our privacy analysis shall nevertheless identify whether finding p, q or r will still remain a hard problem given the availability of n and (U, V, W, R). Clearly, we can no longer consider the problem to be those of pure factorization or pure discrete logarithm.

Firstly we should point out that because r is a random secret and is sufficiently large (review the end of Section 3.3 for the magnitude of r), we can consider that the pairs (U, V) and the pair (W, R) are computationally unrelated. Therefore we can examine their impacts on the privacy of the scheme separately.

Now let's identify the difficulty for finding p, q from n and (U, V). Suppose there

exists an efficient algorithm $\mathcal{A}$ such that with input (f, U, V, n) it will output p and q in time bounded by a polynomial in the size of n. We should keep in mind that $\mathcal{A}$ works because the input values are related by

$$n \equiv \log_f(U) \log_f(V) \,(\mathrm{mod}\, Q), \tag{13}$$

where (review Section 3) Q is the order of f and is a prime. Were the input values not related in any way then because to date there exists no polynomial-time algorithms to factor integers or to compute discrete logarithms, $\mathcal{A}$ should not have output $\log_f(U)$, $\log_f(V)$ in time bounded by any polynomial in the size of n. Notice that Q in the relation in (13) is a prime; so for any $x < Q$, (13) is equivalent to

$$n \equiv [x \log_f(U)][x^{-1} \log_f(V)] \equiv \log_f(U^x) \log_f(V^{x^{-1}}) \,(\mathrm{mod}\, Q).$$

So with input $(f, U^x, V^{x^{-1}}, n)$ $\mathcal{A}$ should output $\log_f(U^x)$ and $\log_f(V^{x^{-1}})$ in time bounded by a polynomial in the size of n. Further notice that U^x (and $V^{x^{-1}}$) forms a permutation in the group generated by f. Thus, given a quadruple (f, U', V', f^n) with U', V' arbitrary elements in the group generated by f, $\mathcal{A}$ forms an efficient decision procedure for deciding whether (f, U', V', f^n) forms a Diffie-Hellman quadruple.

The privacy of r can be argued analogously using the quadruple $(f, W, R, \frac{f^{n+1}}{UV})$.

The above argument indicates that finding p and q from n and (U, V) is likely a problem of deciding the Decision Diffie-Hellman problem in a subgroup of a big prime order. This is widely regarded a hard problem [20, 26].

5 Performance

The proposed scheme consists of (i) two instances of running *Product*, (ii) the proof of the two-prime product structure of n, and (iii) the verifiable secret sharing of r.

Since (iii) should be common to any verifiable threshold secret sharing schemes, we shall only analyze the performance due to (i) and (ii) as additional costs in achieving verifiable partial sharing of integer factors.

In (i), the main cost of running *Product* is to prove the size of two integers (twice running *Bit-Size*), plus a trivial cost for the proof of discrete logarithm equality (a three-move protocol): in each run of *Bit-Size*, Alice and Bob will each compute $2k$ exponentiations mod P where k is the number of iterations in *Bit-Size*; in the proof of discrete logarithm equality they only need to compute a few additional exponentiations mod P. Since on average one exponentiation mod P amounts to $1.5 \log_2 P$ multiplications mod P, the total number of multiplications for two instances of running *Product* can be bounded by $7k \log_2 P$.

Next, we look at the cost for the proof of the two-prime product structure of n. Because n is a Blum integer, the protocol of Van de Graaf and Peralta [12] can be applied which involves agreeing k pairs of random numbers in $(Z/nZ)^*$ and k bits as random signs. For each pair of the agreed numbers, Alice needs to compute a square root which costs an exponentiation mod n; Bob's computation is trivial: to evaluate a Jacobi symbol and to check one squaring; all together his performance is at the level of a few multiplications mod n. Finally, the protocol of Boyar et al for the proof of square-free numbers [5] needs to be run for k times, each run costs one exponentiation mod n for each party. Thus, after k runs of the both protocols we can use $3k \log_2 n$ to bound the total number of multiplications mod n computed by Alice; Bob will need to do no more than half of this amount.

In Lemma 2 we know that the size of n should be three bits smaller than that of Q which can be of similar size of that of P (e.g., let $P = 2Q + 1$). Thus we can use $(\log_2 P) - 4$ to bound the size of n.

Our performance analysis concludes in the following theorem.

Theorem 1 *With a verifiable secret sharing scheme for sharing a secret in the form of a discrete logarithm, the proposed scheme for verifiable partial sharing of the prime factors of an integer of size up to* $(\log_2 P) - 4$ *bits has a cost for both prover and verifier bounded by* $7k \log_2 P$ *multiplications modulo* P *and* $3k \log_2 n$ *multiplications modulo* n. □

Let P a 1540 bit prime; then the maximum size of the integers that can dealt with by the proposed scheme will be 1536 ($= 3 \times 512$) binary bits. Verifiable partial escrow of the prime factors of an integer of this size with a sufficiently small error probability of $1/2^{40}$ can be achieved with no more than 615,520 multiplications.

6 Conclusion

We have constructed a verifiable partial key escrow scheme for integer factoring based cryptosystems. To the author's knowledge this is the first practically efficient construction which features time-delayed and (t, m)-threshold key recovery. We have also analyzed the security of the scheme and our analysis indicates that the difficulty of finding the escrowed secret through an unspecified way is likely to be that of solving a Decision Diffie-Hellman problem in a group of a big prime order. The new scheme realizes a practical method for fine tuning the time complexity for factoring an integer and the tunable complexity has no respect to the size of the integer. The time complexity for factoring an integer using a known factor of $\phi(n)$ has an independent value in that

it should be a piece of must-know knowledge for designing protocols or cryptosystems based on disclosing a factor of $\phi(n)$ of a non-trivial size.

Acknowledgments

I would like to thank the anonymous referees of Designs, Codes and Cryptography for valuable comments and the lead to reference [2], and Nigel Smart for helpful discussions.

References

[1] M. Bellare and S. Goldwasser. Verifiable partial key escrow. Proceedings of 4th ACM Conference on Computer and Communications Security. ACM Press. April 1997. pages 78–91.

[2] M. Bellare and S. Goldwasser. Encapsulated key escrow. MIT Laboratory for Computer Science Technical Report 688, November 1996. Presented at rump session of EUROCRYPT 96, May 1996. Available at `http://www-cse.ucsd.edu/users/mihir/papers/escrow.html`.

[3] B. Blackley. Safeguarding cryptographic keys. Proceedings of the National Computer Conference 1979, volume 48 of American Federation of Information Processing Societies Proceedings, pages 313–317, 1979.

[4] M. Blum. Coin flipping by telephone: a protocol for solving impossible problems. Proceedings of 24th IEEE Computer Conference (CompCon), 1982. pages 133–137.

[5] J. Boyar, K. Friedl and C. Lund. Practical zero-knowledge proofs: Giving hints and using deficiencies. Advances in Cryptology: Proceedings of EUROCRYPT 89 (J.-J. Quisquater and J. Vandewalle, eds.), Lecture Notes in Computer Science 434, Springer-Verlag (1990) pages 155–172.

[6] D. Chaum and T. P. Pedersen. Wallet databases with observers. Advances in Cryptology: Proceedings of CRYPTO 92 (E.F. Brickell, ed.), Lecture Notes in Computer Science 740, Springer-Verlag (1993) pages 89–105.

[7] H. Cohen. *A Course in Computational Algebraic Number Theory.* Springer-Verlag Graduate Texts in Mathematics 138 (1993).

[8] D. Coppersmith. Finding a small root of a bivariate integer equation; factoring with high bits known. Advances in Cryptology — Proceedings of EUROCRYPT 96 (U. Maurer, ed.), Lecture Notes in Computer Science 1070 Springer-Verlag (1996) pages 178–189.

[9] I.B. Damgård. Practical and provably secure release of a secret and exchange of signatures. Advances in Cryptology: Proceedings of EUROCRYPT 93 (T. Helleseth, ed.), Lecture Notes in Computer Science 765, Springer-Verlag (1994) pages 201–217.

[10] M. Girault. An identity-based identification scheme based on discrete logarithms modulo a composite number. Advances in Cryptology: Proceedings of EUROCRYPT 90 (I.B. Damgård, ed.), Lecture Notes in Computer Science 473, Springer-Verlag (1991) pages 481–486.

[11] M. Girault and J.C. Paillès. An identity-based scheme providing zero-knowledge authentication and authenticated key-exchange. First European Symposium on Research in Computer Security – ESORICS 90 (1990) pages 173–184.

[12] J. Van de Graaf and R. Peralta. A simple and secure way to show the validity of your public key. Advances in Cryptology: Proceedings of CRYPTO 87 (E. Pomerance, ed.), Lecture Notes in Computer Science 293, Springer-Verlag (1988) pages 128–134.

[13] S. J. Kim, S. J. Park and D. H. Won. Convertible group signatures. Advances in Cryptology: Proceedings of ASIACRYPT 96 (K. Kim, T. Matsumoto, eds.), Lecture Notes in Computer Science 1163, Springer-Verlag (1996) pages 310–321.

[14] D.E. Knuth. The Art of Computer Programming, Volume 1, Fundamental Algorithms. Addison-Wesley, 1973.

[15] W. Mao. Necessity and realization of universally verifiable secret sharing. 1998 IEEE Symposium on Security and Privacy, IEEE Computer Society (1998) pages 208–214.

[16] W. Mao and C.H. Lim. Cryptanalysis of prime order subgroup of Z_n^*. Advances in Cryptology: Proceedings of ASIACRYPT 98 (K. Ohta, D. Pei, eds.), Lecture Notes in Computer Science 1514, Springer-Verlag (1998) pages 214–226.

[17] A.J., Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.

[18] S. Micali. Fair public key cryptosystems. Advances in Cryptology — Proceedings of CRYPTO'92 (E. F. Brickell, ed.) Lecture Notes in Computer Science 740, Springer-Verlag (1993) pages 113–138.

[19] S. Micali. Guaranteed partial key escrow. MIT/LCS TM-537 September 1995.

[20] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th Annual Symposium on Foundations of Computer Science*, 1997.

[21] P.C. van Oorschot. and M.J. Wiener. Parallel collision search with cryptanalytic applications. *J. of Cryptology*, Vol.12, No.1 (1999), pages 1–28.

[22] T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. Advances in Cryptology: Proceedings of CRYPTO 91 (J. Feigenbaum, ed.), Lecture Notes in Computer Science 576, Springer-Verlag (1992) pages 129–120.

[23] J.M. Pollard. Monte Carlo method for index computation (mod p), *Mth. Comp.*, Vol.32, No.143 (1978), pages 918–924.

[24] A. Shamir. How to share a secret. Communications of the ACM, Vol 22 (1979) pages 612–613.

[25] A. Shamir. Partial key escrow: a new approach to software key escrow. Presented at Key Escrow Conference, Washington, D.C., September 15, 1995.

[26] M. Stadler. Publicly verifiable secret sharing. Advances in Cryptology: Proceedings of EUROCRYPT 96 (U. Maurer, ed.), Lecture Notes in Computer Science 1070, Springer-Verlag (1996), pages 190–199.