Verifiable Partial Sharing of Integer Factors

Wenbo Mao

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

Abstract. It is not known to date how to partially share the factors of an integer (e.g., an RSA modulus) with verifiability. We construct such a scheme on exploitation of a significantly lowered complexity for factoring $n = pq$ using a non-trivial factor of $\phi(n)$.

1 Introduction

Partial key escrow purports to add a great deal of difficulty to mass privacy intrusion which is possible in ordinary key escrow with abusive authorities while preserving the property of an ordinary escrowed cryptosystem in targeting small number of criminals. In partial key escrow, a portion of an individual's private key with an agreed and proved size will not be in escrow. Key recovery requires a non-trivial effort to determine the missing part. A partial key escrow scheme must render that the determination of the missing key part will only be possible *after* recovery of the key part which is in escrow (usually with a set of distributed agents who are collectively trusted to share the escrowed key part). If the missing part can be determined before, or without taking, a prescribed key recovery procedure, then off-line pre-computations can be employed for finding the missing part and this can be done in a massive scale with many or all users targeted. This constitutes a so-called premature key recovery attack: the missing key part is not really missing and the whole private key of each user can be made available right after recovery of the escrowed key part. The effect of partial key escrow is thereby nullified and the scenario of mass privacy intrusion can still be assumed just as the case of an ordinary key escrow scheme. In their recent work "Verifiable partial key escrow", Bellare and Goldwasser [1] discussed scenarios of premature key recovery attacks.

Thus, a necessary step in verifiable partial key escrow is for a key owner to prove that a private key contains a hidden number which will not be in escrow and has an agreed size. To discover this number requires first to recover the escrowed part of the private key, and only after that recovery can an exhaustive search procedure be lunched to determine the missing number. The cost of the search will be a well-understood problem given the proved size of the missing number.

The previous verifiable partial key escrow scheme of Bellare and Goldwasser [1] was proposed for discrete logarithm based cryptosystems. In that realization,

S. Tavares and H. Meijer (Eds.): SAC'98, LNCS 1556, pp. 94–105, 1999.

a key $y = g^x$ is cut into two parts $y_1 y_2 = g^{x_1} g^{x_2} = g^{x_1+x_2}$ where x_1 is a partial private key and is proved in escrow, x_2 is the remaining private component which will not be in escrow but has an agreed bit size to be proved by the key owner during the key escrow time. Using a bit commitment scheme these proofs can be done without revealing the partial private and public keys. Key recovery will have to be via recovering x_1, y_1 then $y_2 = y/y_1$ followed by searching x_2 from y_2. Note that neither of the partial public keys should be revealed before recovering x_1, or else searching x_2 can take place before x_1 is recovered.

It is not known to date how to partially escrow integer factors (e.g., prime factors of an RSA modulus) with verifiability. Full key escrow for integer factoring based cryptosystems can take the approach of escrowing a prime factor of an integer [12] (the scheme in [12] achieves public verifiability). It is however not straightforward to perceive a partial integer-factor sharing scheme along that approach. A major problem is to establish a precise cost for key recovery (e.g., a 2^{40}-level time cost, which is non-trivial but expensively manageable by a well resourced agent, and has been chosen as a suitable workload for the partial key escrow scheme for discrete-log based cryptosystems [1].) Chopping down an 80-bit block[1] from a prime factor and throwing it away will unlikely be a correct rendering because, in integer factoring based cryptosystems, a prime factor should have a size significantly larger than 80 bits, and so the remaining escrowed part of the factor will be sufficiently large to allow exploitation of polynomial-time factoring methods (e.g., Coppersmith [6]) to factor the integer in question. On the other hand, throwing away a much larger number block may render a key unrecoverable.

We will construct a scheme for verifiable partial sharing of the factors of an integer. The central idea in our realization is an observation (one of the cryptanalysis results in [13]) on a significantly lowered and precisely measurable time complexity for factoring $n = pq$ if a factor of $\phi(n)$ (the Euler phi function) is known. This factor will be proved to have an agreed size and will be in escrow in verifiable secret sharing. We will reason that without recovery of this factor, integer factoring will be as the same difficult as the original factoring problem.

In the remainder of the paper, Section 2 describes a lowered complexity for factoring n with a known factor of $\phi(n)$, Section 3 constructs the proposed scheme; Section 4 analyzes the security and performance of the scheme; finally, Section 5 concludes the work.

2 Integer Factoring with an Additional Knowledge

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

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

[1] Searching a number of this size requires 2^{40} operations using known best algorithms based on a square-root reduction; we will discuss this reduction in Section 2.

Let r be a factor of $(p-1)(q-1)$, but not a factor of $p-1$ and $q-1$ (otherwise r^2 will be a factor of $(p-1)(q-1)$). Then

$$p+q \equiv n+1 \,(\mathrm{mod}\, r). \tag{2}$$

When r is smaller than $p+q$, congruence (2) means the equation

$$p+q = (n+1 \mod r) + kr, \tag{3}$$

for an unknown k. If $p+q$ becomes known, factoring n into p and q follows a simple calculation. Equation (3) shows that if r is known, finding the quantity $p+q$ is equivalent to finding the unknown k, and hence the difficulty to factor n is equivalent to that to find k.

Let $|a|$ denote the bit size of the integer a in the binary representation. For $|p+q| > |r|$, noting $(n+1 \mod r) < r$, we have

$$|k| + |r| - 1 \leq |p+q| \leq |k| + |r| + 1,$$

or

$$|k| \approx |p+q| - |r|.$$

So when r is known, the time needed to determine $p+q$, or to factor n, is bounded above by performing

$$2^{|p+q|-|r|} \tag{4}$$

computations. The algorithm is to search k in equation (3).

The bound in (4) can further be lowered significantly. Note that most elements in the multiplicative group of integers modulo n Z_n^* have orders larger than $p+q$ unless factoring n is easy. This means that arbitrarily picking $u \in Z_n^*$, and letting v be the least number satisfying

$$u^v \mod n = 1,$$

then in an overwhelming probability,

$$v > p+q.$$

Combining (1) and (3), we will have

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

Raising u to the both sides of (5), writing $w = u^r \mod n$, and noting $u^{(p-1)(q-1)} \equiv 1 \,(\mathrm{mod}\, n)$, we have

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

We point out that the quantity k in (6) is exactly that in (3) because $kr < p+q$ which should be smaller than the order of u (namely, transforming (3) to (6), the number k will not be reduced in modulo the order of u). With r (hence

w) known, extracting k from the equation (6) using Shank's baby-step giant-step algorithm (see e.g., [5], Section 5.4.1) will only need about

$$2^{\frac{|k|}{2}}$$

operations (multiplication in Z_n^*) and the same order in the space complexity. This is a much lowered bound from (4) as it is the positive square root of (4). To this end we have proven the following statement.

Lemma 1. *Let $n = pq$ for p, q being two distinct primes, and r be a known factor of $\phi(n) = (p-1)(q-1)$ with $|r| < |p+q|$. Then time and space for factoring n can use $2^{\frac{|p+q|-|r|}{2}}$ as an upper bound.* □

A number of previous integer factoring based cryptosystems make use of a disclosed sizable factor of $\phi(n)$ where n is the product two secret primes [8,9,11]. Each of these systems includes moduli settings that allow feasible factorization using our method. For instance, a modulus setting in [8] satisfies $\frac{|p+q|-|r|}{2} = 35$.

Remark Lemma 1 states an upper bound, which means to factor $n = pq$ with a known $r|(p-1)(q-1)$ needs *at most* $2^{\frac{|p+q|-|r|}{2}}$ operations. However, this complexity measurement reaches the lowest known to date on exploitation of computing small discrete logarithms. Any algorithm using fewer operations will provide a breakthrough improvement on solving the (small) discrete logarithm problem. The same argument applies to the verifiable partial key escrow scheme of Bellare and Goldwasser.

If r is unknown, then equation (3), hence equation (6), are not usable. For n being the product of large primes p and q, factorization is so far known to be an infeasible problem. The verifiable partial integer-factor sharing scheme to be specified in the next section will make use of this big time-complexity difference between factoring n with a sizeable factor of $(p-1)(q-1)$, and doing it without.

3 The Proposed Scheme

A user (Alice) shall construct her public key $n = pq$ satisfying that p and q are primes of the same size, $(p-1)(q-1)$ has a factor r with $|p+q| - |r| = 80$ and r is not a factor of both $p-1$ and $q-1$.

Alice shall then prove in zero-knowledge the following things: (i) n is the product of two hidden prime factors p and q of the same size; (ii) a hidden number r is a factor of $(p-1)(q-1)$ but not that of both $p-1$ and $q-1$, satisfying $\frac{|n|}{2} - |r| = 80$; (iii) verifiable secret sharing of r with a set of agents (called shareholders). After all of these have been done, we know from the previous section that if r is recovered by co-operative shareholders, n can be factored by finding k in (6) with 2^{40} operations using the known fastest algorithms in computing small discrete logarithms. On the other hand, if r is not available, n

will be in the same position as an ordinary RSA modulus, and given the size of n, no computationally feasible algorithm is known to factor it.

To this end we have fully described the working principle of a verifiable partial integer-factor sharing scheme. In the remaining of this section we shall construct computational zero-knowledge protocols for 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 [4] (for proof of discrete logarithm equality), Damgård [7] (for proof of integer size), Pedersen [14] (for verifiable threshold secret sharing), and Mao [12] (for 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 $Q = (P-1)/2$ be also prime. Let $f \in Z_P^*$ be a fixed element of order Q. The public group is the multiplicative group generated by f. We assume that it is computationally infeasible to compute discrete logarithms to the base f. Once setup, the numbers f and P (hence $Q = (P-1)/2$) will be announced for use by the system wide entities.

3.1 Proof of Correct Integer Arithmetic

We shall apply $y = f^x \bmod P$ as the one-way function needed to prove the correct integer arithmetic. (In the sequel we will omit the presentation of modulo operations whenever the omission does 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 a verifier (Bob) 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.

However note that for the one-way function f^x used, the proved multiplication relationship depicted above is in terms of modulo $ord(f)$ (here $ord(f) = Q$ is 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 true in the space of integers. Nevertheless, if Alice can show the bit sizes of the respective discrete logarithms (i.e., $|\log_f(A)|$, $|\log_f(B)|$ and $|\log_f(C)|$), then the following lemma guarantees the correct multiplication.

Lemma 2. *Let* $ab = c \,(\mathrm{mod}\, Q)$ *and* $|c| + 2 < |Q|$. *If* $|a| + |b| \leq |c| + 1$ *then* $ab = c$.

Proof. Suppose $ab \neq c$. Then $ab = c + \ell Q$ for some integer $\ell \neq 0$. Noticing $0 < c < Q$, so

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

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

Thus $y = f^x$ forms a suitable one-way function to be used for proof of the correct product of integers in its pre-image space, provided the bit sizes of the

pre-images are shown. Given f^x, there exists efficient protocols to show $|x|$ without revealing x ([7]). In the next subsection we will specify a simplified variation of the proof which protects x up to computational zero-knowledge. Applying the bit-size proof protocol, we can specify a protocol (predicate) $Product(A, B, C)$ below. 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 $Product(f^a, f^b, f^{ab})$
(a run will be abandoned with 0 returned if any party finds any error in any checking step, otherwise it returns 1 upon termination.)
Alice sends to Bob: $|a|$, $|b|$, $|ab|$, and demonstrates the following evidence:

i) $\log_f(f^a) = \log_{f^b}(f^{ab})$ and $\log_f(f^b) = \log_{f^a}(f^{ab})$;
ii) $|a| + |b| \leq |ab| < |Q| - 2$.

In $Product$, showing (i) can use the protocol of Chaum and Pedersen [4], and showing (ii) can use a protocol $Bit\text{-}Size$ 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 (i), the size regarding the third value is always the summation of those regarding the former two.

We will analyze the security and performance of this protocol in Section 4.

3.2 Proof of Integer Size

The basic technique is due to Damgård [7]. We specify a simplified version based on the discrete logarithm problem. In our variation, the number in question is protected in computational (statistical) zero-knowledge.

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

Protocol $Bit\text{-}Size(f^s)$
Execute the following k times:

1. Alice picks $0 < t_1 < e$ at uniformly random, and sets $t_2 := t_1 - e$; she sends to Bob the unordered pair $C_1 := f^{t_1}$, $C_2 := f^{t_2}$;
2. Bob selects $b = 0$ or $b = 1$ at uniformly random, and sends b to Alice;
3. Alice sets $u_1 := t_1$, $u_2 := t_2$ for $b = 0$, $u_1 := t_1 + s$, $u_2 := t_2 + s$ for $b = 1$, and sends u_1, u_2 to Bob;
4. Bob checks the following (with $i = 1, 2$):
   for $b = 0$, $-e < u_i < e$ and $C_i = f^{u_i}$;
   for $b = 1$, $a - e < u_i < b + e$ and $C_i f^s = f^{u_i}$;

Set, for instance, $I := [2^{\ell-1},\ 2^{\ell}]$. Then e will be $2^{\ell-1}$, and $Bit\text{-}Size$ will prove that the discrete logarithm of the input value does not exceed $\ell + 1$ binary bits.

3.3 Proof of the Prime Factors' Structure

Let $n = pq$ be the user Alice's public key for an integer factoring based cryptosystem. The proposed scheme will 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 distinct odd numbers, any two of them are relatively prime, and their sizes satisfy

$$|p'| - |s| = |q'| - |t| = 80. \tag{8}$$

Then set

$$r = 4st. \tag{9}$$

As a procedure for key setup, Alice should first choose the numbers p', q', s, t at random with the required sizes, oddity and relative-prime relationship. She then samples if p and q in (7) are prime. The procedure repeats until p and q are prime. For p', q', s, t being odd, both p and q will be congruent to 3 modulo 4, rendering n to be a Blum integer [2]. We will need this property for proof of two prime structure of n. It is advisable that p' and q' be chosen as primes, resulting in p and q as the so-called strong primes (for n in a secure size, p' and q' will be large). This follows a desirable moduli setting for integer factoring based cryptosystems.

From (7) and (8) we know

$$|p| - 2|s| = |p'| - |s| = 80 = |q'| - |t| = |q| - 2|t|.$$

Noting $|p| = |q|$, so the above implies $|s| = |t|$, and

$$|p + q| - |r| \approx |p| + 1 - |4st| \approx |p| + 1 - (2 + 2|s|) \in \{79, 80, 81\}.$$

Once the key values are fixed, Alice shall publish

$$A = f^p, \quad B = f^q, \quad C = f^r, \quad D = f^{\frac{(p-1)(q-1)}{r}}.$$

Using these published values, Alice can prove to Bob the following two facts:

i) $n = \log_f(A) \log_f(B)$, by running $Product(A, B, f^n)$;

ii) $\log_f(C)$ divides $(\log_f(A) - 1)(\log_f(B) - 1)$, by running $Product(C, D, \frac{f^{n+1}}{AB})$ (note here the dis-log of the third input is $(\log_f(A) - 1)(\log_f(B) - 1)$).

During the proof that runs $Product$, Alice has also demonstrated the bit size values $|\log_f(A)|$, $|\log_f(B)|$ and $|\log_f(C)|$. Bob should check that

$$|\log_f(A)| - |\log_f(C)| \approx \frac{|n|}{2} - |\log_f(C)| \in \{79, 80, 81\}.$$

Finally Alice should prove that n consists of two prime factors only. This can be achieved by applying the protocol of Van de Graaf and Peralta [10] for proof of Blum integers (we have constructed n to be a Blum integer), and the protocol of Boyar et al [3] for proof of square-free integers.

3.4 Verifiable Secret Sharing of r

For self-containment, we include Pedersen's threshold verifiable secret sharing scheme [14] 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 ([15]), 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_Q^*, 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

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

Each shareholder can verify

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

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., [12] 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 Analysis

We now provide security and performance analyzes on the proposed verifiable partial integer-factor sharing scheme.

4.1 Security

The security of the proposed scheme consists of the correctness and the privacy. The former concerns whether Alice can successfully cheat Bob to accept her proof using a key in which the private component is either not recoverable,

or recoverable at a much higher cost than that of the procedure specified in Section 2. The latter concerns whether Bob can gain any knowledge, as a result of verifying Alice's proof, leading to discovery of Alice's private key without going through the prescribed key recovery procedure.

Correctness

In Section 2 we have established the precise cost for key recovery. The remark following Lemma 1 further emphasizes that should there exist an algorithm that can find the missing key part with fewer than 2^{40} operations, that algorithm will also form a breakthrough improvement from all methods we know to date for computing small discrete logarithms. The remaining correctness issue is on that of the sub-protocols which realize the mathematics established in Section 2.

The two sub-protocols that construct *Product* have well understood correctness properties [4,7]. The cheating probability for proof of a Diffie-Hellman triple is $1/Q$ where Q is the order of f, and that for *Bit-Size* is $1/2^k$, here k is the number of iterations in the it. Using $k = 40$ will achieve a sufficiently low chance for successful cheating by Alice. So we can use $1/2^{40}$ as the cheating probability for *Product* (since $1/Q \ll 1/2^{40}$ for usual sizes of Q).

Next, the correctness of the protocol for proof of two-prime structure is also well established [10,3], with error probability $1/2^k$ for k iterations of message verification. Again, k can be set to 40.

We note that in the proofs for the structural validity of n (i.e., $n = pq$, the primality of p, q, the relation $r|(p-1)(q-1)$, and the required sizes of these numbers), the verification job does not involve handling any secret. Therefore it can be carried out by anybody and can be repeated if necessary. So Bob cannot collude with Alice without risking to be caught.

Finally, the correctness of the protocol for verifiable sharing of r [14] has a two-sided error. The error probability for Alice's successful cheating is $1/Q$ which is negligible as Q is sufficiently large. 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). Usually the threshold value t is set to $\lfloor \frac{m}{2} \rfloor$.

Privacy

Although each of the sub-protocols used in the construction of the main scheme has its own proved privacy, we cannot assume that when they run in combination they will still provide a good privacy for the main scheme. Thus, we shall study the privacy of the main scheme in the following manner: (i) the values that have been made public in the main scheme do not themselves leak useful information for finding the prime factors, and (ii) when viewed together, they will not produce a formula usable for finding the prime factors. We emphasize that the quality of (ii) is essential as we must be sure that the published values will not damage the original privacy of each sub-protocol used in the main scheme.

In addition to the public key n, in the main scheme Alice has published the following values:

$$A = f^p,\ \ B = f^q,\ \ C = f^r,\ \ D = f^{\frac{(p-1)(q-1)}{r}},\ \ |p| = |q| = \frac{|n|}{2},\ \ |r| \approx \frac{|n|}{2} - 80.$$

First of all it is obvious that there is no danger for disclosing the bit size information of the secret numbers. So we need only to consider the first four numbers: A, B, C and D.

We note that (A, B, f^n) and $(C, D, \frac{f^{n+1}}{AB})$ forming two Diffie-Hellman triples. This fact, however, will not help Bob to find the discrete logarithms of A, B, C, and D, following the security basis of the Diffie-Hellman problem. Analogously, any arithmetic of these numbers will not produce anything easier than the original Diffie-Hellman problem.

We should remind that, even for n having a widely-believed least size of 512 bits, the discrete logarithms of the published values above will all be sufficiently large. The least one is r whose size satisfies

$$|r| \geq 256 - 81 = 175,$$

which is still sufficiently large and immune to the square-root attack aimed for extracting it from C.

Next we shall review if any of the published numbers will damage the original privacy of each sub-protocol used. We can safely begin without concerning any impact on the two sub-protocols used in proof of two-prime structure of n because those protocols involves calculations in a different group: Z_n^* rather than Z_P^*.

The two sub-protocols that construct *Product* require Alice to hide the private input values (which are discrete logs of the input to *Product* or to *Bit-Size*) by using a genuine random number in each round of message exchange (these random numbers form the prover's random input; review [4] for proof of Diffie-Hellman triple, and Section 3.2 for proof of integer size). In both sub-protocols, the prover's random input will have similar sizes as those of the private input. In fact, the sub-protocol for proof of Diffie-Hellman triple is perfect (computational) zero-knowledge [4], and that for proof of bit size is statistical zero-knowledge [7]. As a result, *Product* can be simulated using the standard way for demonstrating an honest verifier zero-knowledge protocol.

Finally we point out that Pedersen's protocol for verifiable sharing of r (the discrete log of C) has a well understood perfect privacy following the privacy of Shamir's secret sharing scheme based on polynomial interpolation [14,15].

4.2 Performance

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

Since (i) can be viewed as common in verifiable secret sharing schemes, we shall only analyze the performance of (ii) and (iii) as an additional cost for achieving verifiable partial sharing of integer factors.

The major cost in a run of *Product* is to prove the size of two integers (twice running *Bit-Size*), adding a trivial cost for proof of discrete logarithm equality (a three-move protocol). For the two runs of *Product*, total data to be exchanged in *Product* will be bounded above by $(4k+2)|P|$ (binary bits) where k is the number of iterations in *Bit-Size*. The number of computations needed to be performed by both the prover and the verifier can be expressed by $4k|P|$ (multiplications modulo P).

Next, in the proof of two-prime structure of n, the protocol of Van de Graaf and Peralta [10] involves agreeing k pairs of random numbers in Z_n^* and k bits as random signs. For each pair of the numbers agreed, the two parties will evaluate a Jacobi symbol which is at a level of performing a few multiplications in Z_n^*. So the data to be exchanged here will be bounded above by $(2k+1)|n|$ (binary bits), where k can be as the same as that in *Bit-Size*. A similar estimate apply to the protocol for proof of square-free numbers [3]. The number of computations need to be performed by both the prover and the verifier can be expressed by $2k|n|$ (multiplications modulo n). We can use $|P|$ to up-bound $|n|$.

Combining the results in the two paragraphs above, we conclude in following statement for the performance of the proposed scheme.

Lemma 3. *With a verifiable secret sharing scheme for sharing the discrete logarithm of an integer, partial sharing of the prime factors of an integer of size up to* $|P|-3$ *(binary bits) can be achieved by further transmitting* $(8k+3)|P|$ *binary bits, with both the prover and the verifier computing* $6k|P|$ *multiplications.* □

If $|P|$ is set to 1540, then the maximum size of the integers that can dealt with by the proposed scheme will be 1536 (binary bits) $(= 3 \times 512)$. Considering setting $k = 40$ will allow a sufficiently small probability of $1/2^{40}$ for the prover successfully cheating. Then, the total number of bits to be transmitted will be 497,420 (62 kilobytes). A slightly smaller number of multiplications (modulo P and modulo n) will need to be performed by both the prover and the verifier.

5 Conclusion

We have constructed a verifiable partial integer-factor sharing scheme and shown that the scheme is secure and practically efficient. The working principle of the scheme is an observation on a significantly lowered time complexity for factoring $n = pq$ using a known factor of $\phi(n)$. This is of independent interest in that the lowered complexity bound should be regarded as 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 wish to thank the reviewers of the SAC'98 for helpful comments.

References

1. Bellare, M. and S. Goldwasser. Verifiable partial key escrow. Proceedings of 4th ACM Conference on Computer and Communications Security. ACM Press. April 1997. pp. 78–91.
2. Blum, M. Coin flipping by telephone: a protocol for solving impossible problems. Proceedings of 24th IEEE Computer Conference (CompCon), 1982. pp. 133–137.
3. Boyar, J., 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, Springer-Verlag, 434 (1990) pp 155–172.
4. Chaum, D. and T. P. Pedersen. Wallet databases with observers. Advances in Cryptology: Proceedings of CRYPTO 92 (E.F. Brickell, ed.), Lecture Notes in Computer Science Springer-Verlag, 740 (1993) pp. 89–105.
5. Cohen, H. *A Course in Computational Algebraic Number Theory.* Springer-Verlag Graduate Texts in Mathematics 138 (1993).
6. Coppersmith, D. 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, Springer-Verlag, 1070 (1996) pp. 178–189.
7. Damgård, I.B. 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, Springer-Verlag, 765 (1994) pp. 201–217.
8. Girault, M. 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, Springer-Verlag, 473 (1991) pp. 481–486.
9. Girault, M. 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) pp. 173–184.
10. Van de Graaf, J. and R. Peralta. A simple and secure way to show the validity of your public key. Advances in Cryptology: Proceedings of CRYPTO 87 (E. Pomerence, ed.), Lecture Notes in Computer Science, Springer-Verlag, 293 (1988) pp. 128–134.
11. Kim, S. J., 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, Springer-Verlag, 1163 (1996) pp. 310–321.
12. Mao, W. Necessity and realization of universally verifiable secret sharing. 1998 IEEE Symposium on Security and Privacy, IEEE Computer Society (1998) pp. 208–214.
13. Mao, W. and C.H. Lim. Cryptanalysis of prime order subgroup of Z_n^*. Advances in Cryptology: Proceedings of ASIACRYPT 98, Lecture Notes in Computer Science, Springer-Verlag (to appear: November 1998).
14. Pedersen, T. Non-interactive and information-theoretic secure verifiable secret sharing. Advances in Cryptology: Proceedings of CRYPTO 91 (J. Feigenbaum, ed.), Lecture Notes in Computer Science, Springer-Verlag, 576 (1992) pp. 129–120.
15. Shamir, A. How to share a secret. Communications of the ACM, Vol 22 (1979) pp 612–613.