On the Security of a Practical Identification Scheme*

Victor Shoup
Bellcore, 445 South St., Morristown, NJ 07960
shoup@bellcore.com

July 29, 1996

Abstract

We analyze the security of an interactive identification scheme. The scheme is the obvious extension of the original square root scheme of Goldwasser, Micali and Rackoff to 2^mth roots. This scheme is quite practical, especially in terms of storage and communication complexity. Although this scheme is certainly not new, its security was apparently not fully understood. We prove that this scheme is secure if factoring integers is hard, even against active attacks where the adversary is first allowed to pose as a verifier before attempting impersonation.

1 Introduction

An *identification scheme* is an interactive protocol in which one party, P (the prover), tries to convince another party, V (the verifier), of its identity. The prover P has a secret key that allows it, and no one else, to convince the verifier of its identity.

There are several types of attacks that an adversary, trying to impersonate P, may attempt, but we focus on just two basic types. In a *passive attack*, the adversary attempts the impersonation using only his knowledge of the public key. In an *active attack*, the adversary is first allowed to interact with P, perhaps several times, posing as V, but not necessarily following V's protocol. As this type of attack is quite feasible in many practical situations, security against active attacks is clearly preferable to security against only passive attacks.

In this paper, we analyze the security of an identification scheme. The scheme is the obvious extension of the original square root scheme of Goldwasser, Micali and Rackoff [5] to 2^mth roots. This scheme is quite practical, especially in terms of storage and communication complexity. Although this scheme is certainly not new, its security was apparently not fully understood. We prove that—like the square root scheme—this scheme is secure against active attacks if factoring integers is hard.

The square root and 2^mth root schemes

Let us first recall the square root scheme. For a given security parameter k, a secret/public key pair is generated as follows. A modulus n is constructed by multiplying two distinct, randomly

*A preliminary version of this paper appeared in the proceedings of *Eurocrypt '96*.

selected primes, both of binary length k; also, an element $a \in \mathbf{Z}_n^*$ is chosen at random, and we set $b = a^2$. The public key is (b, n), and the secret key is a.

Let m be a parameter such that 2^m grows faster than any polynomial in k. The protocol then repeats the following m times:

1. P chooses $r \in \mathbf{Z}_n^*$ at random, computes $x = r^2$, and sends x to V.

2. V chooses $e \in \{0, 1\}$ at random, and sends e to P.

3. P computes $y = r \cdot a^e$ and sends y to V; V accepts if $y^2 = x \cdot b^e$, and rejects otherwise.

Upon termination, V accepts if it accepted at every iteration, and otherwise rejects.

The results of Goldwasser, Micali, and Rackoff imply that the square root scheme is secure against active attacks if factoring is hard.

We consider the following variant, which might be called the 2^mth root scheme. For security parameter k, a secret/public key pair is generated as follows. A modulus n is constructed by multiplying two distinct, randomly selected primes, both of binary length k, and both congruent to 3 mod 4; also, an element $a \in \mathbf{Z}_n^*$ is chosen at random, and we set $b = a^q$, where $q = 2^m$, and m is a parameter such that 2^m grows faster than any polynomial in k. The public key is (b, n), and the secret key is a.

The protocol then executes the following—just once:

1. P chooses $r \in \mathbf{Z}_n^*$ at random, computes $x = r^q$, and sends x to V.

2. V chooses $e \in \{0, \ldots, q-1\}$ at random, and sends e to P.

3. P computes $y = r \cdot a^e$ and sends y to V; V accepts if $y^q = x \cdot b^e$, and rejects otherwise.

Clearly, the 2^mth root is scheme fairly efficient, and is vastly superior to the square root scheme in terms of communication complexity.

Related Work

There are several other variants of the square root scheme, but the only other one in the literature that is secure against active attacks if factoring is hard is the Feige/Fiat/Shamir (FFS) scheme [3]. The FFS scheme is actually a parameterized family of schemes, one of which is the parallel version of the square root scheme. Compared to the FFS scheme, the 2^mth root scheme has much smaller space and communication complexities, which makes the 2^mth root scheme much more attractive in many practical situations, such as smart card implementations. Ohta and Okamoto [13] discuss variants of the FFS scheme.

The Guillou/Quisquarter (GQ) scheme [8, 9] is the same as the 2^mth root scheme, except that 2^m is replaced by an m-bit prime number. It is only known to be secure against *passive* attacks, provided that RSA-inversion is hard (a possibly stronger assumption than the hardness of factoring). Guillou [7] analyzes a variant of the 2^mth roots scheme where n is the product of two primes, one congruent to 3 mod 8 and the other congruent to 7 mod 8; the public key is always $b = 4$. Guillou shows that this scheme is secure against passive attacks if factoring is hard. The same ideas have been discussed by Micali [12].

The Ong and Schnorr (OS) scheme [15] is also a parameterized family of schemes, one of which is the 2^mth root scheme. Ong and Schnorr give a security analysis against passive attacks only; moreover, the hardness assumption is somewhat contrived. Jakobsson [11] has shown that this scheme is secure against passive attacks if factoring is hard.

Other identification schemes are based on the hardness of the discrete logarithm problem. The scheme of Schnorr [16] is only known to be secure against *passive* attacks, provided the discrete logarithm problem in (a subgroup of) $\mathbf{Z}_p^*$, where p is prime, is hard. Brickell and McCurley [2] give a variant of Schnorr's scheme, and give a security analysis against active attacks; however, the hardness assumption is quite unnatural.

Okamoto [14] gives modifications of the GQ and Schnorr schemes which are proved secure, even against active attacks, under the same intractability assumptions of the corresponding original schemes. These schemes, however, are somewhat more complicated and less efficient than the original schemes.

Identification Schemes vs. Signatures

We note that there are well-known techniques (see, e.g., [1]) using cryptographic hash functions for converting identification schemes into digital signature schemes, in which proofs of security are based on an "ideal" hash function model (the hash function is viewed as a random oracle). In this very powerful (and unrealistic) model, it is generally easier to prove that these signature schemes are secure than it is to prove that the original identification schemes are secure against active attacks. This is because it suffices to prove that the original identification schemes are secure only against honest-verifier attacks, i.e., active attacks in which the adversary does not deviate from V's protocol. Indeed, the results of Jakobsson mentioned above already imply that the 2^mth root scheme (and, more generally, the OS scheme) is secure as a signature scheme in the ideal hash function model if factoring is hard.

Proof techniques

One approach to showing that an identification scheme is secure is to show that it is zero knowledge proof of knowledge. This approach is not at all applicable to proving the security of the 2^mth scheme.

Our proof of security does not show that the 2^mth scheme is zero-knowledge (in the sense of [5]). Indeed, the results of Goldreich and Krawczyk [4] and Itoh and Sakurai [10], together with the result of Jakobsson, imply that there is no "black box" zero-knowledge simulator, assuming factoring is hard.

Perhaps even more interestingly, this scheme is provably not a proof of knowledge (in the sense of [3]), assuming factoring is hard. Indeed, any knowledge extractor could be used to take square roots modulo n as follows. Given a random square $c \in \mathbf{Z}_n^*$, we construct a prover P' with public key $(c^{2^{m-1}}, n)$. Prover P' can make V accept with probability $1/2$, since P' can respond correctly to all even challenges e; the knowledge extractor would hence be obliged to compute, by interacting with P', a square root of c.

As we shall see, this apparent lack of "soundness" is not really a problem at all. In fact, this property is crucial to our proof of security, whose main technical innovation is a "partial zero-knowledge simulation" technique that exploits this property.

Organization of the paper

The rest of the paper is organized as follows: in §2, we formally state our definition of security; in §3, we give a proof of security for the 2^mth root scheme; in §4, we make some concluding remarks, including a discussion of several minor extensions of our results.

2 Definition of Security

In this section, we formally state our definition of a secure identification scheme, which is essentially that of [3]. For conciseness and clarity, we adopt the notation of [6] for expressing the probability of various events. If S is a probability space, then $[S]$ denotes the set of elements in this space that occur with nonzero probability. For probability spaces $S_1, S_2, \ldots$, the notation

$$\Pr[p(x_1, x_2, \ldots) \mid x_1 \leftarrow S_1; x_2 \leftarrow S_2; \ldots]$$

denotes the probability that the relation $p(x_1, x_2, \ldots)$ holds when each x_i is chosen, in the given order, from the corresponding probability space S_i.

A probabilistic algorithm A on a specific input x produces an output string according to some probability distribution. We denote by $A(x)$ the probability space of output strings.

Generally, an *identification scheme* (G, P, V) consists of a probabilistic, polynomial-time algorithm G, and two probabilistic, polynomial-time, interactive algorithms P and V with the following properties.

1. The algorithm G is a *key-generation algorithm.* It takes as input a string of the form 1^k (i.e., k written in unary), and outputs a pair of strings (S, I). k is called the *security parameter*, S is called a *secret key*, and I is called a *public key.*

2. P receives as input the pair (S, I) and V receives as input I. After an interactive execution of P and V, V outputs a 1 (indicating "accept") or a 0 (indicating "reject"). For a given S and I, the output of V at the end of this interaction is of course a probability space and is denoted by $(P(S, I), V(I))$.

3. A legitimate prover should always be able to succeed in making the verifier accept. Formally, for all k and for all $(S, I) \in [G(1^k)]$, $(P(S, I), V(I)) = 1$ with probability 1.

An *adversary* (P', V') is a pair of probabilistic, polynomial-time, interactive algorithms. For a given secret/public key pair (S, I), we denote by $(P(S, I), V'(I))$ the string h output by V' (on input I) after interacting with P (on input (S, I)) some number of times. Note that these interactions are performed sequentially. Again, for a given S and I, $(P(S, I), V'(I))$ is a probability space. The string h (a "help string") is used as input to P', which attempts to make V (on input I) accept.

Definition 1 *An identification scheme* (G, P, V) *is* secure against active attacks *if for all adversaries* (P', V')*, for all constants* $c > 0$*, and for all sufficiently large* k*,*

$$\Pr[s = 1 \mid (S, I) \leftarrow G(1^k);\ h \leftarrow (P(S, I), V'(I));\ s \leftarrow (P'(h), V(I))] < k^{-c}.$$

3 Security of the 2^mth root scheme

Our proof of security is based on the assumption that factoring is hard. We make this assumption precise.

For $k \geq 5$, let H_k be the probability space consisting the uniform distribution over all integers n of the form $n = p_1 \cdot p_2$, where p_1 and p_2 are distinct primes of binary length k, and $p_1 \equiv p_2 \equiv 3 \pmod 4$.

The **Factoring Intractability Assumption (FIA)** is the following assertion:

for all probabilistic, polynomial-time algorithms A, for all $c > 0$, and for all sufficiently large k,

$$\Pr[x \text{ is a nontrivial factor of } n \mid n \leftarrow H_k;\ x \leftarrow A(n)] < k^{-c}.$$

We shall prove:

Theorem 1 *Under the FIA, the 2^mth root scheme is secure against active attacks.*

To prove this theorem, we show that any adversary that succeeds in an impersonation attempt with non-negligible probability can be converted into a probabilistic, polynomial-time factoring algorithm that succeeds with non-negligible probability. This is Lemma 1 below.

Let (P', V') be such an adversary. Then there are polynomials $T_i(k)$, $N_i(k)$, $T_{ol}(k)$, and $T_p(k)$ as follows.

- $T_i(k)$ is a bound on the time required for V' to run the protocol once with P, and includes the computation time of P. This computation is done "on-line," and thus will typically have to be fast.
- $N_i(k)$ is a bound on the number of times V' runs the protocol with P. This will typically be small.
- $T_{ol}(k)$ is a bound on the "off-line" time for V'; i.e., all time spent by V' other than running the protocol with P.
- $T_p(k)$ is a bound on the running-time of P' and V. This computation is also "on-line," and thus will also typically have to be fast.

For a given public key (b, n) and "help string" h, let

$$\epsilon(h, b, n) = \Pr[(P'(h), V(b, n)) = 1],$$

where this probability is taken over the coin tosses of P' and V. Then there exist polynomials $Q_1(k)$ and $Q_2(k)$ and an infinite set $K \subset \mathbf{Z}_{>0}$ such that for all $k \in K$,

$$\Pr[\epsilon(h, b, n) \geq Q_2(k)^{-1} \mid (a, (b, n)) \leftarrow G(1^k);\ h \leftarrow (P(a, (b, n)), V'(b, n))] \geq Q_1(k)^{-1}.$$

Lemma 1 *Assume an adversary as above. Then there is a probabilistic factoring algorithm A that runs in time*

$$O((N_i(k)T_i(k) + T_p(k))Q_2(k) + T_{ol}(k) + k^3)$$

such that for all sufficiently large $k \in K$,

$$\Pr[x \text{ is a nontrivial factor of } n \mid n \leftarrow H_k;\ x \leftarrow A(n)] \geq Q_1(k)^{-1}/8.$$

The special form of integers $n \in [H_k]$ implies that the operation of squaring on $\mathbf{Z}_n^*$ acts as a permutation on the subgroup $(\mathbf{Z}_n^*)^2$ of squares. Also, every square z in $\mathbf{Z}_n^*$ has precisely 4 square roots, exactly one of which is also a square. For convenience, and quite arbitrarily, for $w \in \mathbf{Z}_n^*$ we define $C(w) \in \{1, \ldots, 4\}$ to be the relative position of w among the 4 square roots of w^2 when viewed as integers between 0 and $n-1$.

We now describe and at the same time analyze our factoring algorithm. In all statements concerning probabilities, the underlying probability space consists of the random choice of the input n and the coin tosses of the algorithm.

On input $n \in [H_k]$, the algorithm begins by computing l as the smallest nonnegative integer with $2^l \geq Q_2(k)$. We require that $0 \leq l \leq m-1$, which will hold for all sufficiently large k, since we are assuming that 2^m grows faster than any polynomial in k.

The algorithm runs in three stages, as follows.

Stage 1 *This stage takes as input n, runs in time $O(N_i(k)T_i(k)Q_2(k)+T_{ol}(k))$, and outputs $(\tilde{a}, b, h)$ where $\tilde{a}, b \in \mathbf{Z}_n^*$ with $\tilde{a}^{2^{m-l}} = b$ and h is a "help string." Moreover, we have:*

(i) if $k \in K$, then $\Pr[\epsilon(h, b, n) \geq Q_2(k)^{-1}] \geq Q_1(k)^{-1}/2$;

(ii) the distribution of $C(\tilde{a})$ is uniform and independent of that of (h, b, n).

This stage runs as follows. We choose $\tilde{a} \in \mathbf{Z}_n^*$ at random and compute $b = \tilde{a}^{2^{m-l}}$. Note that the distribution of (b, n) is independent of $C(\tilde{a})$, and is the same as that of an ordinary public key—this is because squaring permutes $(\mathbf{Z}_n^*)^2$, and so b should be, and is, just a random square. We then simulate the interaction $(P(\cdot, b, n), V'(b, n))$. This is complicated by the fact that we do not have at hand a value a such that $a^{2^m} = b$, but rather $\tilde{a}$ with $\tilde{a}^{2^{m-l}} = b$.

We employ a variation of a zero-knowledge simulation technique introduced by Goldwasser, Micali, and Rackoff [5], which might be called partial zero-knowledge simulation. We replace the identification protocol with the following:

1′. P chooses $e_0' \in \{0, \ldots, 2^l - 1\}$ at random, chooses $r \in \mathbf{Z}_n^*$ at random, computes $x = r^{2^m} \cdot b^{-e_0'}$, and sends x to V'.

2′. V' computes a challenge $e \in \{0, \ldots, 2^m - 1\}$ and sends e to P.

3′. P writes $e = e_1 2^l + e_0$. If $e_0 \neq e_0'$, we go back to step 1′ ("undoing" the computation of V'). Otherwise, P computes $y = r \cdot \tilde{a}^{e_1}$ and sends y to V'.

An easy calculation shows that if $e_0 = e_0'$, then $y^{2^m} = x \cdot b^e$; moreover, it is easily seen that the distribution of (x, y) is precisely the same as it would be in the original protocol, and is independent of $C(\tilde{a})$.

The expected number of loop iterations until $e_0 = e_0'$ is 2^l. Over the course of the entire interaction between P and V', the expected total number of challenges is at most $2^l N_i(k)$. If the total number of challenges ever exceeds twice this amount, we quit and output an arbitrary h; otherwise, we output the h that V' outputs. We also output $\tilde{a}$ and b.

That completes the description of Stage 1. All of the claims made above about this stage are easily verified.

Stage 2 *This stage takes as input h, b, and n from above, and runs in time $O(T_p(k)Q_2(k))$. It reports failure or success, and upon success outputs $z \in \mathbf{Z}_n^*$ and $f \in \{0, \ldots, 2^m - 1\}$ such that $z^{2^m} = b^f$ and $f \not\equiv 0 \pmod{2^{l+1}}$. The probability of success, given that $\epsilon(h, b, n) \geq Q_2(k)^{-1}$, is at least $1/2$.*

Let $\epsilon = \epsilon(h, b, n)$, and assume $\epsilon \geq Q_2(k)^{-1}$.

We use a slight variation of an argument in Feige, Fiat, and Shamir [3]. Consider the Boolean matrix M whose rows are indexed by the coin toss string ω of P' and whose columns are indexed by the challenges e of V. $M(\omega, e) = 1$ if and only if this choice of ω and e causes V to accept.

Call a row ω in M heavy if the fraction of 1's in the row is at least $3\epsilon/4$. Observe that the fraction of 1's in the matrix that lie in heavy rows is at least $1/4$. Now consider a heavy row ω, and a challenge e such that $M(\omega, e) = 1$. Consider the fraction of challenges e' with the property that $M(\omega, e') = 1$ and $e' \not\equiv e \pmod{2^{l+1}}$. This fraction is at least

$$3\epsilon/4 - 2^{-l-1} \geq 2^{-l-2} > Q_2(k)^{-1}/8.$$

Stage 2 repeats the following procedure 7 times. Run $(P'(h), V(b, n))$ up to $\lceil Q_2(k) \rceil$ times, or until V accepts. If V accepts, we have a relation $y^{2^m} = xb^e$. Fixing the coin tosses of P', run the interaction again up to $\lceil 8Q_2(k) \rceil$ times, or until V accepts again with a challenge $e' \not\equiv e \pmod{2^{l+1}}$. If V accepts with such a challenge, then we have another relation $(y')^{2^m} = xb^{e'}$. Upon finding two such relations, and ordering them so that $e > e'$, we output $z = y/y'$ and $f = e - e'$.

The probability that an individual execution of the above procedure succeeds is at least $(1 - 1/2.718)^2/4$. By a simple calculation, it follows that the probability that one of the 7 executions succeeds is more than $1/2$. That completes the description and analysis of Stage 2.

Stage 3 of our factoring algorithm is executed only if Stage 2 succeeds.

Stage 3 *This stage takes as input n, the value $\tilde{a}$ from Stage 1, and the values z and f from Stage 2. It runs in time $O(k^3)$. The probability that it outputs a nontrivial factor of n, given that Stage 2 succeeded, is $1/2$.*

This stage runs as follows. We write $f = u2^t$, where u is odd and $0 \leq t \leq l$. Set $w = z^{2^{l-t}}$. We claim that with probability $1/2$, $\gcd(\tilde{a}^u - w, n)$ is a nontrivial factor of n.

To see this, first note that $w^{2^{m-l+t}} = b^{u2^t}$, and hence $w^{2^{m-l}} = b^u$ (as squaring permutes $(\mathbf{Z}_n^*)^2$). We have $(\tilde{a}^u)^{2^{m-l}} = b^u$ and hence $(\tilde{a}^u)^2 = w^2$ (again, squaring permutes $(\mathbf{Z}_n^*)^2$). But since u is odd, and by the independence of $C(\tilde{a})$ and w, $C(\tilde{a}^u)$ has a uniform distribution independent from $C(w)$, and so with probability $1/2$, $\tilde{a}^u \neq \pm w$. In this case, $\gcd(\tilde{a}^u - w, n)$ is a nontrivial factor of n.

That completes the description and analysis of Stage 3.

It follows that for sufficiently large $k \in K$, the overall success probability of our factoring algorithm is at least

$$Q_1(k)^{-1}/2 \times 1/2 \times 1/2 = Q_1(k)^{-1}/8.$$

That completes the description and analysis of our factoring algorithm, and the proof of Lemma 1.

4 Concluding Remarks

We close with the some remarks about our proof of Theorem 1, and discuss several generalizations.

4.1 Constructiveness of the proof

Our proof of Theorem 1 is not constructive, since to build our factoring algorithm, we need not only descriptions of the adversary's algorithms, but also the polynomials $Q_2(k)$ and $N_i(k)$. However, if we allow our factoring algorithm to have just an expected polynomial running-time bound, we can do without this. Getting rid of $N_i(k)$ is trivial. To get rid of $Q_2(k)$, we can generate l at random, where we choose the value $l \geq 1$ with probability 2^{-2l+1}. We then use 2^l in place of $Q_2(k)$ throughout the factoring algorithm. With this modification, the running time bound in Lemma 1 becomes

$$O(N_i(k)T_i(k) + T_p(k) + T_{ol}(k) + k^3),$$

which is an expected running time bound, and the lower bound on the success probability in Lemma 1 becomes

$$Q_1(k)^{-1}Q_2(k)^{-2}/4.$$

4.2 Efficiency of the reduction

In comparison with the reduction from factoring to impersonating one obtains with the FFS scheme, ours is less efficient, since our factoring algorithm has to repeat computations of V' many times. Compensating for this is the fact that the computations that need to be repeated are all "on-line," and so presumably fast.

4.3 Parallel instances of a prover

In our definitions and proofs of security, we do not allow the adversary to interact with several instances of the same prover in parallel. Our proof breaks down in this case, as the time required for the partial zero-knowledge simulation would grow exponentially in the number of parallel instances. It would seem that in many practical situations, such an attack is impossible to mount or at least easy to prevent. In contrast, the proof of security of the FFS remains valid in this context.

4.4 Multiple users

We have stated the protocols and definition of security from the point of view a single user. Consider a a system consisting of many (but polynomial in k) users. Before attempting an impersonation, we allow an adversary to interact arbitrarily with all users in the system in an arbitrary fashion, interleaving communications between users arbitrarily. We also allow an adversary to corrupt any users it wants, obtaining their private keys upon demand. After this interaction, the adversary tries to impersonate a non-corrupted user of its choice. Note that the adversary makes this choice dynamically; if the choice were static, the analysis of the multi-user case would trivially reduce to the single-user case.

To reduce factoring to impersonating in this case, we can use the obvious technique of first picking a user at random and giving them the number we want to factor, and generating ordinary key pairs for all other users. We then hope that the adversary picks the user with our number.

As with the FFS scheme, all users can share the same modulus n in the 2^mth root scheme. However, to reduce factoring to impersonating in this case, we still have to pick a user at random, and give this user $\tilde{a}, b \in \mathbf{Z}_n^*$ with $\tilde{a}^{2^{m-l}} = b$. Again, the other users get ordinary key pairs, and we have to hope that the adversary chooses to impersonate our user. This is unlike the FFS scheme, where every user gets ordinary key pairs, and the impersonation of any user factors n with high probability.

4.5 The OS scheme

Our proof of security against active attacks easily extends to the OS scheme [15].

In this scheme, the modulus n is chosen precisely as before. There are two parameters s and m, chosen so that 2^{sm} grows faster than any polynomial in k. Set $q = 2^m$. A private key consists of a list $a_1, \ldots, a_s$ of randomly chosen elements of $\mathbf{Z}_n^*$; the corresponding public key consists of $b_1, \ldots, b_s \in \mathbf{Z}_n^*$, where $b_i = a_i^q$ for $1 \leq i \leq s$.

The protocol then runs as follows:

1. P chooses $r \in \mathbf{Z}_n^*$ at random, computes $x = r^q$, and sends x to V.
2. V chooses $e_1, \ldots, e_s \in \{0, \ldots, q-1\}$ at random, and sends $e_1, \ldots, e_s$ to P.
3. P computes $y = r a_1^{e_1} \cdots a_s^{e_s}$ and sends y to V; V accepts if $y^q = x b_1^{e_1} \cdots b_s^{e_s}$, and otherwise rejects.

Theorem 2 *Under the FIA, the OS scheme is secure against active attacks.*

The proof is similar to that of Theorem 1; we sketch the differences. In the factoring algorithm, the value l is chosen to be the least nonnegative integer such that $2^{s(l+1)} \geq 2Q_2(k)$. As before, we require that $0 \leq l \leq m-1$, but this will hold for all sufficiently large k.

In Stage 1, for $1 \leq i \leq s$, we choose $\tilde{a}_i \in \mathbf{Z}_n^*$ at random, and compute $b_i = \tilde{a}_i^{2^{m-l}}$. Then we perform the same simulation as in Stage 1, except this time, we have to guess the low-order l bits of each of the s challenges. This slows down the simulation by a factor of $2^{sl} < Q_2(k)$.

Stage 2 is easily modified so as to obtain $z \in \mathbf{Z}_n^*$ and integers $f_1, \ldots, f_s$ such that $z^{2^m} = \prod_i b_i^{f_i}$ and not all f_i are divisible by 2^{l+1}.

In Stage 3, for $1 \leq i \leq s$, write $f_i = u_i 2^{t_i}$, and let $t = \min\{t_i\} \leq l$. Then it is easy to see that with probability 1/2,

$$\gcd(z^{2^{l-t}} - \prod_{i=1}^{s} \tilde{a}_i^{u_i 2^{t_i - t}}, n)$$

is a nontrivial factor of n.

One could modify the above factoring algorithm as follows. Choose an index $g \in \{1, \ldots, s\}$ at random, select $\tilde{a}_g \in \mathbf{Z}_n^*$ at random, and compute $b_g = \tilde{a}_g^{2^{m-l}}$. For $i \neq g$, compute a_i and b_i as specified in the protocol. With this modification, Stage 1 runs as before, except we only have to guess the low-order l bits of e_g. Thus, the simulation is slowed down by only a factor of 2^l. In Stage 3, the probability of success is reduced to $1/(2s)$. In general, this seems like a worthwhile tradeoff.

References

[1] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, 1993.

[2] E. F. Brickell and K. S. McCurley. An interactive identification scheme based on discrete logarithms and factoring. *J. Cryptology*, 5:29–39, 1992.

[3] U. Feige, A. Fiat, and A. Shamir. Zero-knowledge proofs of identity. *J. Cryptology*, 1:77–94, 1988.

[4] O. Goldreich and H. Krawczyk. On the composition of zero-knowledge proof systems. In *Proceedings of the 17th ICALP, Lecture Notes in Computer Science, Vol. 443*, pages 268–282. Springer, 1990.

[5] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18:186–208, 1989.

[6] S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17:281–308, 1988.

[7] L. Guillou. Collision-resistance and zero-knowledge techniques. Manuscript, 1990.

[8] L. Guillou and J. Quisquater. A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In *Advances in Cryptology–Crypto '88*, pages 216–231, 1988.

[9] L. Guillou and J. Quisquater. A practical zero-knowledge protocol fitted to security microprocesors minimizing both transmission and memory. In *Advances in Cryptology–Eurocrypt '88*, pages 123–128, 1988.

[10] T. Itoh and K. Sakurai. On the complexity of constant round ZKIP of possession of knowledge. *IEICE Trans. Fundamentals*, E76-A(1), 1993.

[11] M. Jakobsson. Reducing costs in identification protocols. Manuscript available at `http://www-cse.ucsd.edu/users/markus`, 1992.

[12] S. Micali. An efficient digital signature algorithm provably secure as integer factorization. Manuscript, 1995.

[13] K. Ohta and T. Okamoto. A modification of the Fiat-Shamir Scheme. In *Advances in Cryptology–Crypto '88*, pages 232–243, 1988.

[14] T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *Advances in Cryptology–Crypto '92*, pages 31–53, 1992.

[15] H. Ong and C. Schnorr. Fast signature generation with a Fiat Shamir-like scheme. In *Eurocrypt*, pages 432–440, 1990.

[16] C. Schnorr. Efficient signature generation by smart cards. *J. Cryptology*, 4:161–174, 1991.