J. Cryptology (1999) 12: 247–260

Journal of
CRYPTOLOGY

# On the Security of a Practical Identification Scheme[*]

Victor Shoup
IBM Zurich Research Laboratory,
Saeumerstrasse 4, 8803 Rueschlikon, Switzerland
sho@zurich.ibm.com

Communicated by Ueli Maurer

Received 29 July 1996 and revised 30 June 1998

**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 $2^m$th 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.

**Key words.** Identification scheme, Proof of security, Zero knowledge.

## 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. Corresponding to its public key, $P$ has a secret key that allows it, and no one else, to convince the verifier of its identity.

A typical application would be an identity card containing a microprocessor (a so-called "smart card"); the secret key would be stored inside the memory of the microprocessor, hidden from view. For example, one could easily imagine passports enhanced in this way.

There are different notions of security for identification schemes. All such notions share a common definition of what it means to "break" the scheme; namely, that an adversary succeeds in an impersonation attempt (making the verifier accept with non-negligible probability). Where these notions of security differ is the type of "attack" the adversary is allowed to mount; that is, how the adversary is allowed to interact with the system before an impersonation attempt.

[*] A preliminary version of this paper appeared in the *Proceedings of Eurocrypt* '96; this work was done while the author was a Humboldt fellow at the University of Saarland, Germany.

The weakest form of attack is a *passive attack*, where the adversary does not interact with the system at all before attempting an impersonation; the only information the adversary has is the public key of the prover.

The strongest form of attack is an *active attack*, where the adversary interacts with $P$, perhaps several times, posing as $V$, but not necessarily following $V$'s protocol.

Active attacks are quite feasible in practice; for example, in the passport example above, a hostile government or corrupt border patrol may require that one's passport be "verified," where the "verification device" actually interacts with the passport, extracting information that allows a forged passport to be constructed.

There are also other attacks of intermediate severity. In an *eavesdropping attack*, the adversary passively eavesdrops on the communication between $P$ and a correct, or honest, $V$; in an *honest-verifier attack*, the adversary plays the role of $V$, and correctly follows $V$'s protocol. For all the identification schemes discussed or mentioned in this paper, these attacks are equivalent to a passive attack, since the verifiers use only "public coins" and the protocols are zero-knowledge against honest verifiers.

The main contribution of this paper is to analyze the security of an identification scheme. The scheme is the obvious extension of the original zero-knowledge proof system for square roots of Goldwasser et al. [8] to $2^m$th 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.

## 1.1. *The Square Root and* $2^m$th *Root Schemes*

We 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 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 analysis of Goldwasser et al. 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 "$2^m$th root scheme." For security parameter $k$, a secret/public key pair is generated as follows. A modulus $n$ is chosen as in the square root scheme. 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 $k$ (we also assume throughout this paper that $m = O(k)$). The public key is $(b, n)$, and the secret key is $a$. We view the value $m$ as an implicit parameter (alternatively, one can make it a part of the public key).

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 $e \in \{0, \dots, q-1\}$ 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 $2^m$th root scheme is fairly efficient, and is vastly superior to the square root scheme in terms of communication complexity.

Our main result is to show that the $2^m$th root scheme is secure against active attacks if factoring is hard. Our proof will require that the prime factors of $n$ are both equal to 3 mod 4 (so that $n$ is a so-called "Blum integer").

This restriction to Blum integers is not really a serious restriction. Indeed, if an adversary can break our scheme, this implies an efficient algorithm that factors a nonnegligible fraction of Blum integers. However, Blum integers make up a nonnegligible fraction of all integers that are the product of two primes of equal binary length. Nevertheless, one can ask if this restriction is absolutely necessary. Schnorr [19] has examined extensions to our results for non-Blum integers; however, the proof in [19] contains a substantial gap, which is addressed in [20].

## 1.2. *The Feige/Fiat/Shamir Scheme*

In the Feige/Fiat/Shamir (FFS) scheme [4], a modulus $n$ is chosen as in the square root scheme. There are two parameters $s$ and $m$, chosen so that $2^{sm}$ grows faster than any polynomial in $k$. A private key consists of a list $a_1, \dots, a_s$ of randomly chosen elements of $\mathbf{Z}_n^*$; the corresponding public key consists of $b_1, \dots, b_s \in \mathbf{Z}_n^*$, where $b_i = a_i^2$ for $1 \le i \le s$.

The protocol then executes the following steps $m$ times *in parallel*:

1. $P$ chooses $r \in \mathbf{Z}_n^*$ at random, computes $x = r^q$, and sends $x$ to $V$.
2. $V$ chooses $e_1, \dots, e_s \in \{0, \dots, q-1\}$ at random, and sends $e_1, \dots, e_s$ to $P$.
3. $P$ computes $y = ra_1^{e_1} \cdots a_s^{e_s}$ and sends $y$ to $V$; $V$ accepts if $y^q = xb_1^{e_1} \cdots b_s^{e_s}$, and otherwise rejects.

Feige et al. prove that this scheme is secure against active attacks if factoring is hard.

Actually, the above scheme is a somewhat simplified version of the FFS scheme, and is closely related to the scheme proposed earlier by Fiat and Shamir [5]. Setting $m = 1$, one obtains a parallel version of the square root scheme, which appeared earlier in a slightly different context in [6]. Ohta and Okamoto [15] discuss variants of the FFS scheme.

Compared with the FFS scheme, the $2^m$th root scheme has much smaller space and communication complexities, which makes the $2^m$th root scheme much more attractive in many practical situations, such as smart card implementations.

## 1.3. *The Ong/Schnorr Scheme*

In the Ong/Schnorr (OS) scheme [17], the modulus $n$ is chosen precisely as in the square root and FFS schemes. 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 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 $e_1, \ldots, e_s \in \{0, \ldots, q-1\}$ 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.

Setting $s = 1$, one obtains the $2^m$th root scheme.

Ong and Schnorr give a security analysis against passive attacks only; moreover, the hardness assumption is somewhat contrived. Jakobsson [13] has shown that this scheme is secure against passive attacks if factoring is hard.

Our proof technique also extends to the OS scheme (again, assuming $n$ is a Blum integer).

## 1.4. *The Guillou/Quisquarter Scheme*

The Guillou/Quisquarter (GQ) scheme [12], [11] is the same as the $2^m$th 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 [10] analyzes a variant of the $2^m$th roots scheme where $n$ is the product of two primes, one congruent to 3 mod 8 and the other congruent to 7 mod 8, and $b = 4$. Guillou shows that this scheme is secure against passive attacks if factoring is hard. The same ideas have been discussed by Micali [14].

## 1.5. *Other Schemes*

Other identification schemes are based on the hardness of the discrete logarithm problem. The scheme of Schnorr [18] 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 [3] give a variant of Schnorr's scheme, and give a security analysis against active attacks; however, the hardness assumption is quite unnatural.

Okamoto [16] 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.

## 1.6. *Identification Schemes versus Signatures*

As was noted by Fiat and Shamir [5], three-round identification schemes such as those described above can be "collapsed" into signature schemes by making the verifier's random challenge equal to the output of a hash function applied to the concatenation of the message to be signed and the first message sent by the prover. Such heuristic arguments can be made in a somewhat disciplined manner by viewing the hash function as a "random oracle," as suggested by Bellare and Rogaway [2]. To make this heuristic argument go through, it suffices to prove that the original identification schemes are secure only against honest-verifier attacks.

Indeed, the results of Jakobsson mentioned above already imply that the $2^m$th root scheme is secure against honest-verifier attacks, and thus as a signature scheme in the ideal hash function model (all this assuming that factoring is hard).

## 1.7. *Proof techniques*

One approach to show that an identification scheme is secure is to show that it is a zero-knowledge proof of knowledge. This approach is not at all applicable to proving the security of the $2^m$th scheme, as we now explain.

Our proof of security does not show that the $2^m$th scheme is zero-knowledge (in the sense of [8]). Indeed, the results of Goldreich and Krawczyk [7], together with the result of Jakobsson, imply that there is no "black box" zero-knowledge simulator, assuming factoring is hard. The argument for this is as follows: the results in [7] essentially say that any efficient black box simulator for a three round, public coin proof system can be turned into a prover that succeeds with nonnegligible probability. Their theorem is stated somewhat differently (since they are concerned with proofs of language recognition, rather than proofs of knowledge), but their argument is easily adapted. Then Jakobsson's result implies such a prover can be used to factor.

We should point out that there are a number of techniques, based on commitment schemes, for transforming an honest-verifier zero-knowledge protocol into a zero-knowledge protocol. These transformations increase the computation cost of the protocol, and can also require additional complexity assumptions to obtain a proof of security. While one could apply these transformations to the $2^m$th root scheme, our results imply that this is unnecessary.

It is perhaps also interesting to note that the $2^m$th root scheme is provably not a proof of knowledge (in the sense of [4]), assuming factoring is hard. Indeed, any knowledge extractor could be used as follows to take square roots modulo $n$ (which is, of course, equivalent to factoring $n$). 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 a technical aside, we note that there are several different notions of a "proof of knowledge" in the literature (as discussed in [1]), but in the case where a prover succeeds as above with probability $1/2$, these notions are in fact all equivalent.

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

The intuition of the proof is as follows. We want to use an adversary that breaks the system to factor a number $n$, which we do as follows. We choose a very small value $l$ (the correct value depends on the exact probability with which the adversary breaks the system). Then we generate a public key $(b, n)$, where $b = \tilde{a}^{2^{m-l}}$ and $\tilde{a} \in \mathbf{Z}_n^*$ is chosen at random. Although to the adversary this looks like a real public key, we do not actually hold a corresponding private key, just an "approximate" private key, or witness $\tilde{a}$—the smaller $l$ is, the better the approximation. The trick then is to show that

(1) despite having just an approximate witness, we can nevertheless efficiently simulate the adversary's view as it would be if it interacted with a prover with a real witness;
(2) if the adversary can subsequently pretend to be a prover and make a verifier accept, then we can extract another approximate witness that is at least as good as the one we started with; and
(3) with these two different approximate witnesses, we can factor $n$.

### 1.8. *Organization of the Paper*

The rest of the paper is organized as follows: in Section 2 we formally state our definition of security; in Section 3 we give a proof of security for the $2^m$th root scheme; in Section 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 [4]. For conciseness and clarity, we adopt the notation of [9] 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. We denote by $(P'(h), V(I))$ the output of $V$ after interacting with $P'(h)$.

**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 $2^m$th 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 algorithms $A$ that run in expected polynomial time, 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* $2^m$th *root scheme is secure against active attacks.*

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

Let $(P', V')$ be such an adversary. For this adversary, there exist polynomials $T_i(k)$, $N_i(k)$, $T_{\text{ol}}(k)$, 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 $P$.
- $N_i(k)$ is a bound on the number of times $V'$ runs the protocol with $P$.
- $T_{\text{ol}}(k)$ 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 $V$.

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

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

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

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

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

$$O((N_i(k)T_i(k) + T_p(k))Q_2(k) + T_{\text{ol}}(k))$$

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

$$\Pr[x \text{ is a nontrivial factor of } n \mid n \leftarrow H_k;\ x \leftarrow A(n)] \geq Q_1(k)^{-1}/4.$$

The special form of integers $n \in [H_k]$ implies that the operation of squaring on $\mathbf{Z}_n^*$ acts as an automorphism on the subgroup $(\mathbf{Z}_n^*)^2$ of squares, which we shall exploit numerous times.

We will need the following notation: let $\chi$ be the natural map from $\mathbf{Z}_n^*$ to $\mathbf{Z}_n^*/(\mathbf{Z}_n^*)^2 \cong \mathbf{Z}_2 \times \mathbf{Z}_2$. Note that, for random $a \in \mathbf{Z}_n^*$, revealing $a^2$ reveals no information about $\chi(a)$; that is, the distributions of $a^2$ and $\chi(a)$ are independent.

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. In the first stage we generate a public key $(b, n)$ with a corresponding "approximate" private key, or witness $\tilde{a} \in \mathbf{Z}_n^*$, where $b = \tilde{a}^{2^{m-l}}$. In this stage we also simulate the view that the adversary would have if it interacted with a proving holding a "real" witness. In the second stage we let the adversary attempt to make a legitimate verifier accept, and use a "rewinding" argument to extract another "approximate" witness that is at least as good as the one we started with. In the third stage we use the two approximate witnesses, and a kind of witness indistinguishability property, finally to factor $n$.

**Stage 1.** This stage takes as input $n$, runs in expected time

$$O(N_i(k)T_i(k)Q_2(k) + T_{\text{ol}}(k)),$$

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 $\Pr[\varepsilon(h, b, n) \geq Q_2(k)^{-1}] \geq Q_1(k)^{-1}$;
(ii) the distribution of $\chi(\tilde{a})$ 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 $\chi(\tilde{a})$, 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 et al. [8]. 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 $V'$.

When the adversary terminates, we output the string $h$ that $V'$ outputs, along with $\tilde{a}$ and $b$.

Notice that in step 1′, the distribution of $x$ is uniformly distributed in $(\mathbf{Z}_n^*)^2$, and its distribution is independent of everything else in the adversary's view up to that point, and is also independent of the hidden variable $e_0'$. Therefore, up to this point the simulation is perfect, and moreover, the probability that $e_0 = e_0'$ is precisely $1/2^l$. Now if it happens that $e_0 = e_0'$, then

$$y^{2^m} = (r\tilde{a}^{e_1})^{2^m} = (xb^{e_0})b^{2^l e_1} = xb^e.$$

Also notice that $x$ leaks no information about $\chi(r)$ or $\chi(\tilde{a})$, and from the equation $\chi(y) = \chi(r)\chi(\tilde{a})^{e_1}$, the distribution of $\chi(y)$ is uniform and independent of $\chi(\tilde{a})$. Thus, the simulation of the adversary's view is perfect through step 3′, and no information is leaked about $\chi(\tilde{a})$. From the fact that $e_0 = e_0'$ with probability $1/2^l$, the expected value of the total number of loop iterations is $2^l N_i(k)$.

All of the claims made about Stage 1 should now be clear.

**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 $\varepsilon(h, b, n) \geq Q_2(k)^{-1}$, is at least 1/2.

Let $\varepsilon = \varepsilon(h, b, n)$, and assume $\varepsilon \geq Q_2(k)^{-1}$. Recall that $l$ was chosen above such that $2^l \geq Q_2(k)$.

Stage 2 repeats the following procedure seven times. Run $(P'(h), V(b, n))$ up to $\lceil Q_2(k) \rceil$ times, or until $V$ accepts. If $V$ accepts, let $y^{2^m} = xb^e$ be the accepting conversation. Fixing the coin tosses of $P'$, run the interaction again up to $\lceil 4Q_2(k) \rceil$ 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 accepting conversation $(y')^{2^m} = xb^{e'}$. Upon finding two such relations, and ordering them so that $e > e'$, we output $z = y/y'$ and $f = e - e'$.

To analyze Stage 2, we use a slight variation of an argument in [4]. Consider the Boolean matrix $M$ whose rows are indexed by the coin toss string $\omega$ of $P'$ and whose columns are indexed by the challenges $e$ of $V$. $M(\omega, e) = 1$ if and only if this choice of $\omega$ and $e$ causes $V$ to accept.

Call a row $\omega$ in $M$ *heavy* if the fraction of 1's in the row is at least $3\varepsilon/4$. Observe that the fraction of 1's in the matrix that lie in heavy rows is at least $1/4$. This follows from the following argument: if $M$ has $r$ rows and $c$ columns, and $r'$ rows are nonheavy, then the total number of 1's in $M$ is $rc\varepsilon$, of which less than $r'c3\varepsilon/4 \leq (3/4)rc\varepsilon$ lie in nonheavy rows.

Now consider a heavy row $\omega$, and a challenge $e$ such that $M(\omega, e) = 1$. Since $2^{m-l-1}$ challenges $e'$ satisfy $e' \equiv e \pmod{2^{l+1}}$, the fraction of challenges $e'$ with the property that

$$M(\omega, e') = 1 \qquad \text{and} \qquad e' \not\equiv e \pmod{2^{l+1}}$$

is at least

$$3\varepsilon/4 - 2^{-l-1} \geq 3Q_2(k)^{-1}/4 - Q_2(k)^{-1}/2 = Q_2(k)^{-1}/4.$$

To complete the analysis of this stage, we make use of the simple fact that if an experiment that succeeds with probability $\theta$ is repeated at least $t$ times, then the probability that all of the experiments fail is at most

$$(1-\theta)^t \leq \exp(-t\theta).$$

Thus, if $t \geq 1/\theta$, the probability that at least one experiment succeeds is at least $1 - \exp(-1)$.

Now, with probability at least $(1 - \exp(-1))$, we hit an accepting pair $(w, e)$. When this happens, the coin tosses of $P'$ index a heavy row with probability $1/4$. Conditioning on the event that we are in a heavy row, we hit the desired $e'$ with probability at least $(1-\exp(-1))$. Thus, the probability that an individual execution of the above procedure succeeds is at least

$$(1 - \exp(-1)) \times 1/4 \times (1 - \exp(-1)) = (1 - \exp(-1))^2/4.$$

By a simple calculation, it follows that the probability that one of seven executions succeeds is more than $1/2$.

All the claims made for Stage 2 should now be clear.

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 $O(mk^2)$. The probability that it outputs a nontrivial factor of $n$, given that Stage 2 succeeded, is $1/2$.

Recall that

$$\tilde{a}^{2^{m-l}} = b, \qquad z^{2^m} = b^f, \qquad \text{and} \qquad f \not\equiv 0 \pmod{2^{l+1}}.$$

Moreover, $\chi(\tilde{a})$ is uniformly distributed and independent of the distribution of $z$.

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 $(\mathbf{Z}_n^*)^2$). However, since $u$ is odd, $\chi(\tilde{a}^u) = \chi(\tilde{a})$, and hence $\chi(\tilde{a}^u)$ is random and independent of $\chi(w)$. It follows that, with probability 1/2, $\tilde{a}^u \neq \pm w$, and when this happens $\gcd(\tilde{a}^u - w, n)$ is a nontrivial factor of $n$.

All the claims made for Stage 3 should now be clear.

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

$$Q_1(k)^{-1} \times 1/2 \times 1/2 = Q_1(k)^{-1}/4.$$

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 polynomial $Q_2(k)$. To eliminate $Q_2(k)$, we can generate $l$ at random, where we choose the value $l \geq 1$ with probability $2^{-l}l^{-2}$, appropriately normed. It is straightforward to verify that we can sample $l$ from this distribution in expected constant time, given a source of random bits. 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

$$O(N_i(k)T_i(k) + T_p(k) + T_{\text{ol}}(k)),$$

and the success probability in Lemma 1 is bounded from below by a constant times

$$1/(Q_1(k)Q_2(k)(\log Q_2(k))^2).$$

### 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 in most realistic attacks.

### 4.3. *Parallel Execution of the Protocol*

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 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 scheme remains valid in this context.

### 4.4. *Multiple Users*

We have stated the protocols and definition of security from the point of view of a single user. Consider 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 noncorrupted user of its choice. Note that the adversary makes this choice dynamically; if the choice were static, the analysis of the multiuser 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 $2^m$th root scheme. However, to reduce factoring to impersonating in this case, we still have to pick a user at random, and give this user an appropriate “approximate witness.” Again, the other users get ordinary key pairs, and we have to hope that the adversary chooses to impersonate our user. The reduction from factoring to impersonating is still polynomial time, but somewhat less efficient than the reduction for the FFS scheme.

### 4.5. *The OS Scheme*

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

We recall the OS scheme described in the Introduction. In this scheme the modulus $n$ is chosen precisely as in the $2^m$th root scheme. 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 $a_1, \dots, a_s$ of randomly chosen elements of $\mathbf{Z}_n^*$; the corresponding public key consists of $b_1, \dots, b_s \in \mathbf{Z}_n^*$, where $b_i = a_i^q$ for $1 \le i \le s$.

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 $e_1, \dots, e_s \in \{0, \dots, q-1\}$ at random, and sends $e_1, \dots, e_s$ 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.

In the analysis of this scheme, we further presume that $n$ is chosen as a Blum integer, just as we did for the $2^m$th root scheme.

**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 $2^{s(l+1)} \ge 2Q_2(k)$. As before, we require that $0 \le l \le m-1$, which holds for all sufficiently large $k$.

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

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

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 O. Goldreich. On defining proofs of knowledge. In *Advances in Cryptology—Crypto* ’92, pages 390–420. LNCS 740. Springer-Verlag, Berlin, 1993.

[2] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

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

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

[5] A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Advances in Cryptology—Crypto* ’86, pages 186–194. LNCS 263. Springer-Verlag, Berlin, 1987.

[6] M. J. Fischer, S. Micali, and C. Rackoff. A secure protocol for the oblivious transfer. *J. Cryptology*, 9(3):191–195, 1996. Presented at Eurocrypt ’84, but not published in those proceedings.

[7] O. Goldreich and H. Krawczyk. On the composition of zero-knowledge proof systems. In *Proceedings of the* 17*th ICALP*, pages 268–282. LNCS 443. Springer-Verlag, Berlin, 1990.

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

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

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

[11] 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. LNCS 330. Springer-Verlag, Berlin, 1998.

[12] L. Guillou and J. Quisquater. A “paradoxical” identity-based signature scheme resulting from zero-knowledge. In *Advances in Cryptology—Crypto* ’88, pages 216–231. LNCS 403, Springer-Verlag, Berlin, 1990.

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

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

[15] K. Ohta and T. Okamoto. A modification of the Fiat-Shamir Scheme. In *Advances in Cryptology—Crypto* ’88, pages 232–243. LNCS 403. Springer-Verlag, Berlin, 1990.

[16] T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *Advances in Cryptology—Crypto '92*, pages 31–53. LNCS 740. Springer-Verlag, Berlin, 1993.

[17] H. Ong and C. Schnorr. Fast signature generation with a Fiat–Shamir-like scheme. In *Advances in Cryptology—Eurocrypt '90*, pages 432–440. LNCS 473. Springer-Verlag, Berlin, 1991.

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

[19] C. Schnorr. Security of $2^t$-root identification and signatures. In *Advances in Cryptology—Crypto '96*, pages 143–156. LNCS 1109. Springer-Verlag, Berlin, 1996.

[20] C. Schnorr. Erratum: Security of $2^t$-root identification and signatures. In *Advances in Cryptology—Crypto '97*. LNCS 1294. Springer-Verlag, Berlin, 1997.