

Reducing costs in identification protocols

Markus Jakobsson
Department of Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093

Abstract

In this paper a protocol for any two users to prove their identities to each other is proposed. The focus of this paper is to minimize the communication overhead and the key lengths without reducing the security. Our protocol achieves this goal by generalizing the Feige, Fiat, Shamir identification protocol [1]. We prove it is secure based on the intractability of factoring random Blum integers.

1 Introduction

In a zero-knowledge identity-proof, one person wants to prove to another person that he is who he claims to be. This must be done without giving away any secret information to the verifier, or somebody that is eavesdropping, so that nobody shall be able to pretend to be him with a greater probability than they had before listening to his proof.

This is a problem first studied by Feige, Fiat and Shamir [1], who constructed the first practical identification protocol. Goldwasser, Micali and Rackoff [2] introduced the notion of zero-knowledge, and Goldreich, Micali and Wigderson [3] produced a general but somewhat inefficient solution to the problem.

The security of an identification protocol is the probability that a polynomial-time adversary who falsely attempts to identify himself gets caught. To reach a specified degree of security, there are four costs for an identification protocol; the cost of communication (how much information must be exchanged), the number of rounds of communication, the cost of memory (how long public keys must be stored) and the cost of computation. In this paper we will be mostly concerned with communication costs and memory. Often, in practice, communication costs outweigh the cost of computation. The cost of memory might also be important, since each potential participant will need to store the public key of all other participants.

In this paper, we introduce a 2-round protocol based on the intractability of factoring Blum integers. The costs for the security $1 - \frac{1}{2^{na}}$ will be $O(n \log N)$ as the cost of memory, $O(na + \log N)$ for communication and $O(na)$ arithmetic operations for computation, where a and n are integer constants arbitrarily chosen, and N is a suitably chosen Blum integer. Here, we assume that N cannot be factored in time $\text{poly}(2^{na} + \log N)$ with a non-negligible probability. All the costs above will be minimized for $n = 1$, but there might be circumstances where larger n 's are beneficial, for example in parallel implementations.

We will describe the protocol, prove that it is complete and sound, show that we can obtain arbitrary security and prove that the protocol is zero-knowledge.

Thus, we can obtain an arbitrary security with zero-knowledge in an arbitrary number of communication steps, where the tradeoff for high security in a few number of communication steps is more local calculation.

Our protocol can be viewed as a variation of Ohta and Okamoto's protocol [4]; however, a much better bound on the security is obtainable in the case we consider.

2 The protocol

All operations are modulo N , where $N = pq$ and p and q are primes such that $p, q \equiv 3 \pmod{4}$. Let Z_N^* be the multiplicative group of integers modulo N . Let QR_N be the set of elements in Z_N^* that are squares. Let a and n be fixed positive integer constants. The security and efficiency of the protocol depend directly on a and n .

A trusted party, such as the protocol designer, chooses the number N as above. All participants know N , but none knows the factorization of N . This allows all participants to use the same value of N . Each (honest) participant chooses a secret identity, a set of numbers $\{s_i\}_1^n$ where each $s_i \in QR_N$, and publishes the set $\{y_i\}_1^n$, where $y_i = s_i^{2^a}$, as her *public identity*.

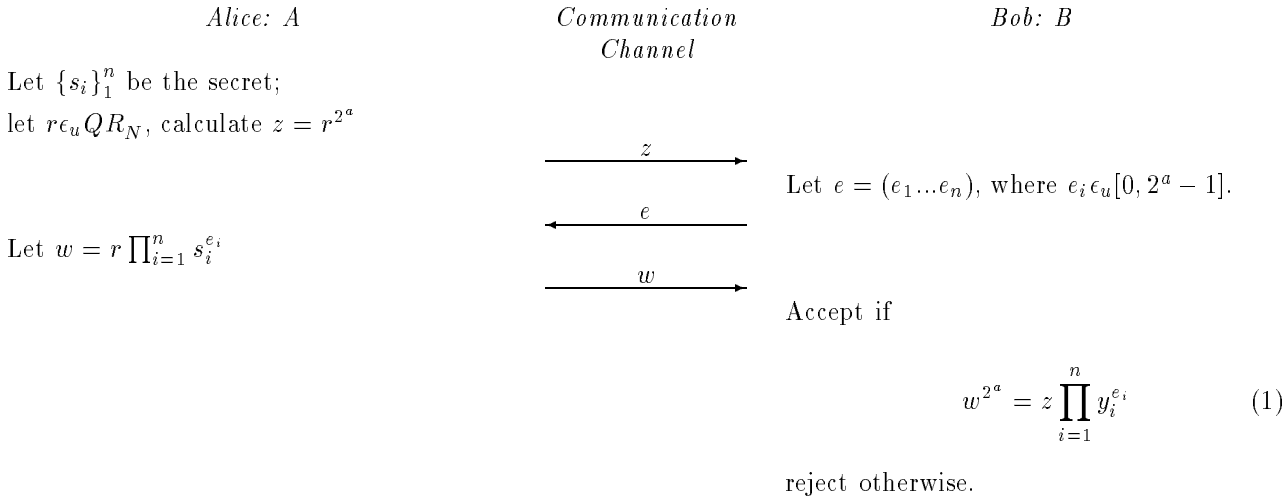
Unlike the FFS [1] protocol, our protocol does not provide a “proof of knowledge” of the s_i ’s, the 2^a ’th roots of the y_i ’s. This is because, while the implementation requires the prover to know the 2^a ’th roots, we prove only that a succesful cheater “knows” the following type of secret: Any root of a product of y_i ’s. However, we show that obtaining such a secret is as hard as factoring N .

For S a finite set, and x a random variable on S , let $x \in_u S$ mean that x is uniformly distributed on S .

Let A be a participant with a secret and B a participant who wants to verify A ’s secret.

A shall prove to B with arbitrary security that she knows the secret, and the proof shall be zero-knowledge. Neither A nor B knows the factorization of N , and they are both computationally polynomially bounded.

The protocol is as follows:



□

Notice that the verifier can start calculating $z \prod_{i=1}^n y_i^{e_i}$ directly after having fixed the e_i ’s, and later just compare it to w^{2^a} to save time. In an implementation of the protocol, B can decrease the number of times to square each side in the verification step by $\min(e_i)$, but to make the proof easier we will not do that here.

The security will be $1 - \frac{1}{2^{na}}$, both the cost of memory $O(n \log N)$, communication $O(na + \log N)$, and that of computation $O(na M(N))$, where $M(N)$ is the time to multiply numbers modulo N . The Feige, Fiat, Shamir identification protocol is a special case of our protocol where $a = 1$. The advantage of our generalization is to allow reduction of communication and memory costs. Memory might be a real obstacle in implementing the FFS scheme, since the scheme requires one to store many keys (integers *mod N*) for *each person* one might interact with.

As in FFS, we can extend this to arbitrary security by repeating the above protocol several times.

Theorem 1 *Assume factoring Blum integers is hard, i.e. no polynomial time probabilistic adversary can factor random Blum integers of length k in time $\text{poly}(2^{n_a} + k)$ with more than a $\frac{1}{\text{poly}(2^{n_a} + k)}$ chance of success.*

The above protocol with N randomly chosen is zero-knowledge to an adversary with time $\text{poly}(2^{n_a} + \log N)$ and if A and B are honest participants, no adversary running in time $\text{poly}(2^{n_a} + \log N)$ can identify herself as A to B with probability more than $\frac{1}{2^{n_a}} + \frac{1}{\text{poly}(2^{n_a} + k)}$

The proof of theorem 1 is found in sections 3 and 4.

Theorem 2 *Let A and B be honest participants in the protocol, let \bar{A} be an adversary with computational power $\text{poly}(2^{n_a} + \log N)$ who interacts with A and B for $\text{poly}(2^{n_a} + \log N)$ different identifications (perhaps using different aliases), and observes A and B interacting for the same number of identifications. Then the probability that \bar{A} can identify herself as A to B in an identification following the above is at most $\frac{1}{2^{n_a}} + \frac{1}{\text{poly}(2^{n_a} + k)}$.*

This theorem follows from theorem 1, and some well-known properties of zero-knowledge. Proof is omitted from this abstract.

3 Completeness and soundness

Obviously the protocol is complete, since A can perform all the calculations needed. The protocol also is sound if B can be sure that no \bar{A} can make him believe she knows the secret, without actually knowing it. This requirement is met if nobody can simultaneously know the answer w to two different queries e and e' without knowing the secret. This is the same as saying that anybody who knows the answers to two queries knows the secret. We show first that if \bar{A} can make B accept her as A with probability significantly larger than $\frac{1}{2^{n_a}}$, then he can compute a number u and a set $S \subseteq \{1, \dots, n\}$, so that $u^2 = \prod_{i \in S} y_i$. We then show that finding such a pair (u, S) is as difficult as factoring N .

Lemma 1 *If $p, q = 3 \pmod 4$, every square mod pq has exactly one square root, which itself is a square.*

Proof of Lemma 1: See reference [5]. ■

Corollary 1 *For any $i \geq 1$, let $x = r^{2^i} \pmod N$ where $r \in_{\mathcal{U}} Z_N^*$. Then $x \in_{\mathcal{U}} QR_N$.*

Lemma 2 *If (z, e, w) and (z, e', w') are both accepted transcripts, then a square root of $(\prod_{i=1}^n s_i^{e_i - e'_i})^2$ can be computed in polynomial time, given only w, w' and N as input.*

Proof of Lemma 2:

Since w is accepted by Bob after (z, e) , and w' is accepted by him after (z, e') ,

$$w^{2^a} = z \prod_{i=1}^n y_i^{e_i}$$

and

$$w'^{2^a} = z \prod_{i=1}^n y_i^{e'_i}$$

Then,

$$w^{2^a} (w'^{2^a})^{-1} = \prod_{i=1}^n y_i^{e_i - e'_i} = \left(\prod_{i=1}^n s_i^{e_i - e'_i} \right)^{2^a}$$

Thus, by repeated application of Lemma 1,

$$(ww'^{-1})^2 = \left(\prod_{i=1}^n s_i^{e_i - e'_i} \right)^2 \quad (2)$$

So the procedure calculates and outputs $t = ww'^{-1}$.

■

Lemma 3 *Let (z, e, w) and (z', e', w') be two accepted transcripts, where $e \neq e'$. We can then calculate a square root of a product of a subset of the y_i 's, i.e., an $S \subseteq \{1..n\}$, $S \neq \emptyset$; and u such that $u^2 = \prod_{i \in S} y_i$.*

Proof of Lemma 3:

Let $f(i)$ denote $e_i - e'_i$. Let (α_i, β_i) , where $\alpha_i = 2^{p_i}$, if $f(i) \neq 0$ be the solution with the smallest possible value of p_i to $f(i) \equiv 2^{p_i} - 2^a \beta_i \pmod{N}$, and let $\alpha = \max(\alpha_i)$. Note that α_i and β_i can be computed easily from the inputs. If we denote $k = \frac{w}{w'}$ and $k_i = s_i^{f(i)}$, then equation 2 implies that $(k^\alpha)^2 = \left(\prod_{i|\alpha_i=\alpha} k_i^\alpha \right)^2 \left(\prod_{i|\alpha_i \neq \alpha} k_i^\alpha \right)^2$. This is, from the definition of α_i , β_i and k_i the same as $\left(\prod_{i|\alpha_i=\alpha} s_i^{2^{a-1}} y_i^{\beta_i} \right)^2 \left(\prod_{i|\alpha_i \neq \alpha} y_i^{\frac{\alpha}{2\alpha_i}} \right)^2$. Note that if $\alpha_i \neq \alpha$, then $2\alpha_i \mid \alpha$, so $\frac{\alpha}{2\alpha_i}$ is an easily computable integer. Therefore, we can calculate a root of $\left(\frac{k^\alpha}{\prod_{i|\alpha_i=\alpha} y_i^{\beta_i} \prod_{i|\alpha_i \neq \alpha} y_i^{\frac{\alpha}{2\alpha_i}}} \right)^2 = \left(\prod_{i|\alpha_i=\alpha} s_i^{2^{a-1}} \right)^2 = \prod_{i|\alpha_i=\alpha} y_i$.

This we will show is the same as knowing a secret or being able to factor N .

■

Lemma 4 *It is hard to calculate a product of roots, i.e. there is no poly-time algorithm, A_{root} , that on input $(y_1^2, y_2^2, \dots, y_n^2)$ for $y_i^2 \in_u QR_N$, with probability greater than $\frac{1}{n^d}$, for any integer d , outputs a pair (S, u) , such that $S \subseteq \{1..n\}$, $S \neq \emptyset$; and u such that $u^2 = \prod_{i \in S} y_i$.*

Proof of Lemma 4:

Assume that there is a poly-time algorithm, A_{root} , that on input $(y_1^2, y_2^2, \dots, y_n^2)$ for $y_i^2 \in_u QR_N$, with probability greater than $\frac{1}{n^d}$, outputs a pair (S, u) , such that $S \subseteq \{1..n\}$, $S \neq \emptyset$; and u such that $u^2 = \prod_{i \in S} y_i$. Then there is a poly-time algorithm that with probability greater than $\frac{1}{n^d}$, given $y^2 \in QR_N$ outputs r such that $r^2 = y^2$. This follows from the fact that if we wanted to calculate a square root of y^2 , we would choose $k \in_u \{1..n\}$, set $y_k^2 = y^2$, and then choose $y_i \in_u Z_N^*$ for $i \neq k$. We would feed $\{y_1^2 \dots y_n^2\}$ to A_{root} and hope that $k \in S$. If $k \in S$, which would happen with a probability larger than $\frac{1}{n^{d+1}}$, we would have calculated a square root of y^2 , namely $u \left(\prod_{i \neq k, i \in S} y_i \right)^{-1}$, in polynomial time. This is known to be as hard as factoring N , and so from our assumption that factoring Blum integers is hard, it follows that the poly-time algorithm A_{root} can not exist.

■

Lemma 5 Assume factoring Blum integers is hard, i.e. no polynomial time probabilistic adversary can factor random Blum integers of length k in time $\text{poly}(2^{n^a} + k)$ with more than a $\frac{1}{\text{poly}(2^{n^a} + k)}$ chance of success.

In the above protocol, with N randomly chosen, with the honest participants A and B , no adversary running in time $\text{poly}(2^{n^a} + \log N)$ can identify herself as A to B with probability more than $\frac{1}{2^{n^a}} + \frac{1}{\text{poly}(2^{n^a} + k)}$

Proof of Lemma 5:

Assume \overline{A} could successfully identify herself to B with probability greater than $\frac{1}{2^{n^a}} + \frac{1}{\text{poly}(2^{n^a} + k)}$, then with a probability larger than $\frac{1}{\text{poly}(2^{n^a} + k)}$, \overline{A} could answer two queries e and e' after sending the message z . From lemmas 2, 3 and 4, this would give us a poly-time algorithm to factor N , contrary to the assumption.

■

4 The protocol is zero-knowledge

Intuitively, the protocol is zero-knowledge if a malicious adversary can not get any more information from the protocol than from a simulation of the protocol that he runs himself. We refer to GMR [2] for a formal definition.

Lemma 6 There is a probabilistic poly-time algorithm *Blackbox* that takes (y, e) as input and produces a string (z, e, w) as output, where this output is distributed identically to the set of transcripts of the protocol with input y , given that B 's response is fixed to the value e .

Proof of Lemma 6: The algorithm *Blackbox*:

input (y, e) , output (z, e, w)

Let $r \in_u QR_N$. Let $w = r$, $z = r^{2^a} (\prod_{i=1}^n y_i^{e_i})^{-1}$.

Then, by definition, $r \in_u QR_N$, and so $w \in_u QR_N$. From Corollary 1, and since as well a power of a quadratic residue as an inverse of a quadratic residue are quadratic residues, it follows that $z \in QR_N$. In the real protocol, $w = r \prod_{i=1}^n s_i^{e_i}$, and since as well s_i as r are quadratic residues, it follows from the above rule that $w \in_u QR_N$. We see that this holds for z , too, as in the real protocol we have $z = r^{2^a}$; therefore $z \in QR_N$ in the real protocol.

From *Blackbox* we get $w \in QR_N$, $z \in QR_N$, and we see that the pair (w, z) will have just the same distributions as in the real protocol. This follows from the fact that (w, z) obeys equation 1. Given that the pair (w, z) obeys equation 1, the distribution on w determines that of z . This follows from Lemma 1 and the fact that $z \in QR_N$.

■

Lemma 7 The protocol is (perfect) zero-knowledge, i.e., there is a probabilistic poly-time algorithm *Simulator* that takes (y, \overline{B}) as input (where \overline{B} is some protocol for a B rather than some fixed answer) and produces a string (z, e, w) that is distributed identically to the set of transcripts determined by a run of the identification protocol between A and \overline{B} on input y . *Simulator* will terminate in expected time $O(2^{n^a} P)$, where P is the time taken by \overline{B} .

Proof of Lemma 7:

The simulator works in the following way: Input to our algorithm *Simulator* is (y, \overline{B}) . *Simulator* guesses a value for e , \overline{e} , and sends this together with y to *Blackbox*. *Blackbox* then outputs (z, \overline{e}, w) , and *Simulator*

checks whether the protocol \overline{B} 's value for e , when it has been given z , would have been \overline{e} . If that is the case, *Simulator* outputs (z, e, w) and halts, otherwise it tries again.

The intuition as to why this works is that \overline{e} was obviously the correct guess of e . According to Lemma 6, the distribution on both w and z will be the same for *Simulator* as in the real protocol, given that $e = \overline{e}$. The output from *Simulator* will therefore have exactly the same distribution as the set of transcripts, where \overline{B} would be used as B 's protocol. Then, the output from *Simulator* will look the same as that from the real protocol, and since Bob can not learn Alice's secret or parts thereof from his own simulation, neither can he from the real protocol, which must therefore be zero-knowledge.

As we will show, if \overline{e} is the simulator's guess of e , then $Prob(\overline{e} = e) = \frac{1}{2^{na}}$. The expectation of the number of tries until *Simulator* terminates is 2^{na} , so it terminates in expected time $O(2^{na}P)$.

Since \overline{e} is chosen uniformly at random from a set of size 2^{na} , it suffices that there is no dependency between \overline{e} and e for $Prob(\overline{e} = e) = \frac{1}{2^{na}}$ to hold. The choice of e depends solely on the value of z . By the definition of the simulator, when \overline{e} is fixed, we have :

$$z = r^{2^a} (\prod_{i=1}^n y_i^{\overline{e}_i})^{-1}.$$

We claim that any value of z can be obtained from any value for \overline{e} , and with equal probability, and so there is no dependency between \overline{e} and e .

Rewrite the above to

$$r^{2^a} = z (\prod_{i=1}^n y_i^{\overline{e}_i}),$$

let $x = z (\prod_{i=1}^n y_i^{\overline{e}_i})$; note that $x \in QR_N$.

Then, for any value of \overline{e} , we must be able to obtain any value in QR_N to compensate, with our choice of r , for all possible \overline{e}_i 's to get our fixed z .

By repeated application of Lemma 1, we see that there is exactly one $r \in QR_N$ such that $x = r^{2^a}$, and so for every \overline{e} there is an r that will give us the right z .

We now will prove that the distribution on \overline{e} given that the simulator halts is identical to that of e in the original protocol. By Lemma 6, this implies that the distributions on triples (w, \overline{e}, z) output by the simulator is identical to the distributions (w, e, z) from the real protocol.

We show that for each fixed random tape for Bob, e and \overline{e} have the same conditional distribution. When the tape is fixed, e is a function of z , $e = e(z)$. Since we only output if we picked $\overline{e} = e(z)$, the distributions (w, \overline{e}, z) and (w, e, z) are identical.

■

5 Acknowledgements

I would like to thank Russell Impagliazzo and Heather Woll for their insightful conversations and support.

6 References

1. U. Feige, A. Fiat, A. Shamir, "Zero-knowledge Proofs of Identity", Journal of Cryptology, 1988, Vol 1, pp 77-94
2. S. Goldwasser, S. Micali, C. Rackoff, "The Knowledge Complexity of Interactive Proof Systems", Proceedings of STOC, 1985, pp 291-304
3. O. Goldreich, S. Micali, A. Wigderson, "Proofs that Yield Nothing but their Validity and a Methodology of Cryptographic Protocol Design", Proceedings of FOCS, 1986, pp 174-187

4. K. Ohta, T. Okamoto, "A Modification of the Fiat-Shamir Scheme", Advances of Cryptology - Crypto '88, Lecture Notes in Computer Science 403, 1988, pp 232-243
5. L. Blum, M. Blum, M. Shub, "A Simple Unpredictable Pseudo-Random Number Generator", SIAM J. COMPUT, Vol 15, No 2, May 1986, p 373, Lemma 1