Zero Knowledge Interactive Proofs of Knowledge
(A Digest)

Martin Tompa

IBM Research Division
Thomas J. Watson Research Center
P. O. Box 218
Yorktown Heights, New York 10598

Lure

Suppose an associate handed you a 500 digit number N, and informed you, "I know the prime factorization of N." What would convince you of the truth of your associate's statement?

If your associate could be persuaded to reveal the factorization to you, a few simple tests would convince you of the statement's truth. Unfortunately the associate responds to this request by saying, "The factorization is a secret. In fact, I would like to convince you that I know the factorization of N without divulging any other useful information." How can you hope to be convinced that your associate is not deceiving you? Needless to say, a primality testing algorithm quickly reveals N to be composite, but your favorite factorization algorithms make no progress whatever.

These seemingly irreconcilable positions (the associate's unwillingness to reveal any knowledge, your unwillingness to accept your associate's statement without proof) are reconcilable through a protocol known as a "zero knowledge interactive proof", introduced by Goldwasser, Micali, and Rackoff [15] in 1985. Informally, an interactive proof is a pair of protocols executed by two parties, called the "prover" and the "verifier", whereby the prover attempts to convince the verifier of the validity of some proposition Π. The prover, even by deviating from its protocol, should not be able to convince the verifier of the truth of Π if, in fact, Π is false. An interactive proof is "zero knowledge" if the verifier, even by deviating from its protocol, cannot gain any information from the prover (other than the validity of Π) that it could not have derived efficiently itself. More specifically, for any verifier that outputs after interacting with the prover, there is an algorithm that, without benefit of interacting with the prover, produces outputs from a distribution indistinguishable from that of the verifier.

The interested reader can find careful definitions of these notions in [20]. The particular problem of knowledge of factorization will be left on the hook until the last section. The intervening sections contain some interesting historical digressions.

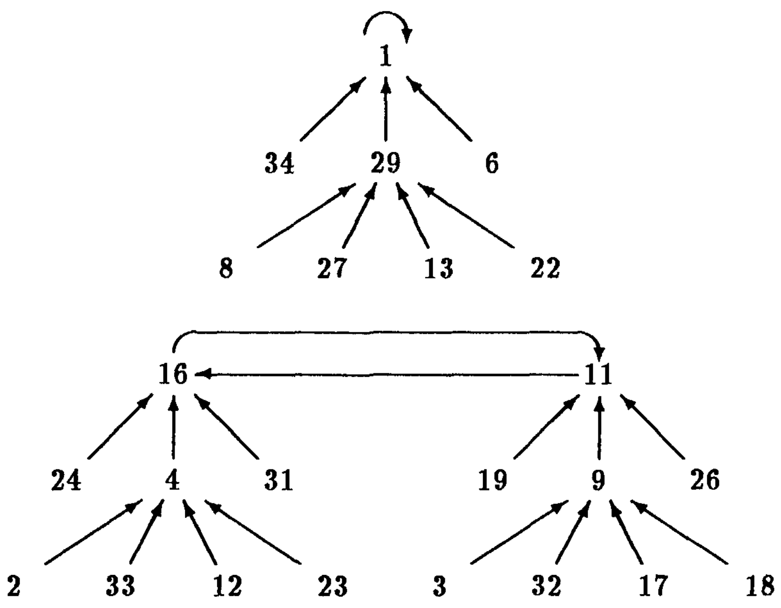

Figure 1: The Structure of Squaring in $\mathcal{Z}_{35}^*$

1 Background in Computational Number Theory

This section discusses some algorithmic questions related to the structure of squaring modulo N, and their relation to factoring. For a more comprehensive introduction to computational number theory, see Angluin [3].

For any positive integer N, let $\mathcal{Z}_N^*$ denote the multiplicative group of integers modulo N, that is,

$$\mathcal{Z}_N^* = \{x \mid 0 < x < N \text{ and } \gcd(x, N) = 1\}.$$

($\gcd(a, b)$ denotes the greatest common divisor of a and b.) Figure 1 contains a directed graph that illustrates the structure of squaring in $\mathcal{Z}_{35}^*$. There is a vertex for each element of $\mathcal{Z}_{35}^*$, and an edge (u, v) whenever $u^2 \equiv v \pmod{35}$. A *quadratic residue modulo N* is simply a "perfect square" in the group $\mathcal{Z}_N^*$. For example, an examination of Figure 1 reveals that the quadratic residues modulo 35 are 1, 4, 9, 11, 16, and 29. Notice also that every quadratic residue modulo 35 has exactly 4 square roots modulo 35, and that they come in 2 pairs of additive inverses modulo 35. (For example, $8 + 27 \equiv 13 + 22 \equiv 0 \pmod{35}$.) The fact that each quadratic residue has 4 square roots modulo 35 derives from the fact that 35 has 2 distinct odd prime factors. In general, if N has k distinct odd prime factors, then every quadratic residue modulo N will have exactly 2^k square roots modulo N.

A function will be said to be *easy* if there is a probabilistic, polynomial expected time algorithm that computes it. (A "probabilistic" algorithm is one that is permitted access to a random number generator. A "polynomial time" algorithm is one that, given any input x, terminates within a number of steps that is a polynomially bounded function of the length of x; for number-theoretic algorithms, integers can be assumed to be input in decimal representation. The expectation in the definition of "easy" is taken over possible outcomes of the random number generator, not over possible inputs.) The following number-theoretic problems, among numerous others, are not believed to be easy [2,4,21]:

Quadratic residuosity: Given N and $x \in \mathcal{Z}_N^*$, determine whether of not x is a quadratic residue modulo N.

Square root: Given N and a quadratic residue x modulo N, output any y such that $y^2 \equiv x \pmod{N}$.

Factorization: Given N, output its prime factorization.

The main result needed from this section is that the last two of these problems are computationally equivalent, in the sense that, if either is easy, the other is easy as well. The remainder of this section outlines the proof of this equivalence. For one half of the equivalence, if you had the prime factorization $p_1^{e_1} p_2^{e_2} \dots p_k^{e_k}$ of N, you could compute a square root of x modulo N by computing a square root of x modulo $p_i^{e_i}$ for each $1 \le i \le k$ (an easy problem due to the special form of the modulus [1,5,17,18]), and combining these results via the Chinese remainder algorithm [16].

The other half of the equivalence is more relevant to subsequent sections. The special case of $k = 2$ distinct prime factors is due to Rabin [17], and will be assumed here for illustrative purposes. The generalization to arbitrary composites N is relatively straightforward [20].

First, note that knowing two square roots modulo N of the same element, one from each of the two pairs of additive inverses, is sufficient to factor N. In particular, if $s^2 \equiv t^2 \pmod{N}$ and $s \not\equiv \pm t \pmod{N}$, then $g = \gcd(s+t, N)$ is a proper factor of N: surely g is a factor of N, so all that remains is to show that $g \ne 1$ and $g \ne N$. The fact that $s + t \not\equiv 0 \pmod{N}$ rules out the possibility $g = N$. Note that N divides $s^2 - t^2 = (s+t)(s-t)$, by hypothesis. If $g = \gcd(s+t, N) = 1$, then N would divide $s - t$, contradicting the hypothesis $s \not\equiv t \pmod{N}$.

To illustrate this fact from Figure 1, 33 and 23 are both square roots of 4 modulo 35, and $\gcd(33+23, 35) = 7$, a proper factor of 35. Similarly, $\gcd(27+13, 35) = 5$, another proper factor.

Now suppose you had an efficient algorithm that produced a single square root of x modulo N, even for only a fixed fraction of the quadratic residues x modulo N. Then you could factor N efficiently as follows. Use the random number generator to find a random, uniformly distributed element t of $\mathcal{Z}_N^*$. Let $x = t^2 \bmod N$, and invoke the hypothesized algorithm to produce a square root s of x modulo N. If this algorithm fails because x is not among the fraction of quadratic residues it can handle, or if $s \equiv \pm t \pmod{N}$, then try a new random t. Otherwise $\gcd(s + t, N)$ is a proper factor of N. Having been given only x, the square root subroutine has no bias toward either pair of square roots. Hence, the expected number of iterations of this procedure until N is factored is constant.

2 Flipping a Coin by Telephone

Zero knowledge interactive proofs can be motivated by an interesting problem known as "flipping a fair coin by telephone" [6]. This problem might arise during a phone call with your boss:

Boss: "I've chosen you to prepare this year's departmental budget."

You: "Why me? Can't you get someone else to do it?"

Boss: "Tell you what: I'll flip a coin. If you call it, I'll find someone else to do the budget."

You: "But, but, ..."

Boss: "Quick, it's in the air. Call it."

You: "Uh, heads."

Boss: "Sorry, it's tails. Have the budget on my desk in the morning."

Rabin [19] and Blum [6], suspecting the possibility of cheating in such a scenario, devised a clever solution based on the presumed intractability of factoring. The protocol for each participant is given in Figure 2. The convention regarding announcement of cheating is that any participant who correctly catches the other cheating automatically wins the toss.

If both participants follow their protocols, then $s \equiv \pm t \pmod{N}$ with probability exactly $\frac{1}{2}$, resulting in a fair coin toss.

It is not difficult to see that the boss can no longer cheat to advantage. Even if s is not uniformly distributed among the square roots of x, the fact that t is so distributed and is secret from the boss prevents any bias of s toward $\pm t \bmod N$. If the boss stupidly chooses N with $k > 2$ distinct odd prime factors, then the probability that $s \equiv \pm t \pmod{N}$ is decreased to $2^{-k+1} \leq \frac{1}{4}$.

Your Private Computation	*Phone Line*	*Boss's Private Computation*
		choose 2 distinct large primes p, q; $N \leftarrow pq$;
	$\xleftarrow{N}$	

if N is a prime power
  then announce cheating;
choose $t \in \mathbb{Z}_N^*$ randomly and uniformly;
$x \leftarrow t^2 \bmod N$;		

	$\xrightarrow{x}$	
		if x is not a quadratic residue modulo N **then** announce cheating; **choose** a random, uniform square root s of x modulo N, using N's factorization as in Section 1;
	$\xleftarrow{s}$	

if $s^2 \not\equiv x \pmod N$
  then announce cheating;
$g \leftarrow \gcd(s + t, N)$;		

	$\xrightarrow{g}$	

	if g is a proper divisor of N
	  then you win
	  else boss wins.	

Figure 2: Protocol for Flipping a Coin by Telephone

Similar reasoning would indicate that you can't cheat your boss to much advantage either. By the presumed intractability of factorization, the chance of you factoring N without the boss's aid can be ignored as negligible. The only aid your boss might be supplying is the random square root s. In the event that you are destined to lose due to $s \equiv \pm t \pmod N$, this is no aid in factoring N, since you could have computed it unaided from t.

Fischer (see [4]) pointed out a subtle flaw in this reasoning. Namely, s might be an aid in factoring N *if you didn't know the value of* t. Suppose there was some hypothetical quadratic residue x that was easy to compute from N, and such that knowledge of any one square root of x modulo N was sufficient to make factoring N easy. Then you could cheat by computing and transmitting such an x, and using the obligingly revealed square root s to factor N. No one knows of the existence of such hypothetical quadratic residues, but no one knows how to prove their nonexistence either. Without the latter, the protocol in Figure 2 cannot be proved fair.

Is the problem of flipping a fair coin by this method doomed? The fortunate answer in the negative came from Goldwasser, Micali, and Rackoff [15], who invented zero knowledge interactive proofs. This idea can be used to circumvent Fischer's objection, as follows [11,15]. After receiving x, the boss should be unwilling to reveal a square root without some convincing "interactive proof" that *you already know some square root* t *of* x *modulo* N. You, on the other hand, want this interactive proof to reveal "zero knowledge" about the value of t, since any such knowledge might enable the boss to bias the choice of s toward $\pm t$ mod N. Such a subprotocol for knowledge of a square root, devised in the same paper by Goldwasser, Micali, and Rackoff [15], should be inserted between the transmissions of x and s. This subprotocol is presented in Section 3. The correctness of the resulting coin-flipping protocol is proved by Fischer *et al.* [12].

3 Proof for Knowledge of a Square Root

Figure 3 contains the protocol for a zero knowledge interactive proof that the prover knows a square root t of x modulo N. A complete proof of the correctness of this protocol is given by Tompa and Woll [20]. The underlying intuition is sketched in the remainder of this section.

If the prover does know some square root of x modulo N and both participants follow their protocols, then neither will announce cheating, independent of the random number generators' output.

Next it will be shown that the probability is at most $\frac{1}{2}$ that the prover doesn't know a square root of x modulo N (and possibly deviates from its protocol), yet the verifier doesn't detect cheating. Although $\frac{1}{2}$ might not seem to be a reassuring level of confidence, the protocol of Figure 3 can simply be repeated τ times to decrease the

Prover (x, N, t)	*Comm. Channel*	*Verifier* (x, N)
comment: $x \equiv t^2 \pmod{N}$;		
choose $r \in \mathcal{Z}_N^*$ randomly and uniformly;		
$y \leftarrow r^2 \bmod N$;		
	$\xrightarrow{y}$	
		choose $\beta \in \{0,1\}$ randomly and uniformly;
	$\xleftarrow{\beta}$	
if $\beta \notin \{0,1\}$ **then** announce cheating;		
$z \leftarrow t^\beta r \bmod N$;		
	$\xrightarrow{z}$	
		if $z^2 \not\equiv x^\beta y \pmod{N}$ **then** announce cheating.

Figure 3: Zero Knowledge Interactive Proof of Knowledge of a Square Root

probability of undetected cheating from $\frac{1}{2}$ to 2^{-r}.

Fix y. For $i \in \{0,1\}$, let z_i be the response sent by the prover to the message $\beta = i$. If the verifier doesn't announce cheating after receiving z_0, then z_0 is a square root of y modulo N. If the verifier doesn't announce cheating after receiving z_1, then z_1 is a square root of xy modulo N. But if the prover knew these two square roots, then the prover would also know a square root $z_0^{-1}z_1 \bmod N$ of x. (This gives an indication of what it means for the prover to "know" a square root, namely, the prover could compute it efficiently. For a careful definition, see [20].) Hence, if the prover doesn't know such a square root of x modulo N, the verifier will announce cheating following one of the two possible choices of β.

Finally, why does the verifier, even by deviating from its protocol, gain only knowledge that it could have computed easily itself? Intuitively, the reason is as follows. When $\beta = 0$, the verifier receives the information $(y, z) = (\rho^2, \rho)$, where ρ is uniformly chosen from $\mathcal{Z}_N^*$. When $\beta = 1$, the verifier receives the information $(y, z) = (x\rho^2, x\rho)$, where ρ is uniformly chosen from $\mathcal{Z}_N^*$. In either case, this is information from a distribution that the verifier can reproduce without the prover's aid.

Prover $(N, \text{factorization}(N))$	*Comm. Channel*	*Verifier* (N)
		choose $t \in Z_N^*$ randomly and uniformly;
		$x \leftarrow t^2 \bmod N$;
	$\xleftarrow{x}$	
Use factorization(N) to compute a square root s of x modulo N, as in Section 1;		
	$\xrightarrow{s}$	
		if $s^2 \not\equiv x \pmod{N}$ **then** announce cheating.

Figure 4: First Interactive Proof of Knowledge of Factorization

Prover $(N, \text{factorization}(N))$	*Comm. Channel*	*Verifier* (N)
		choose $t \in Z_N^*$ randomly and uniformly;
		$x \leftarrow t^2 \bmod N$;
	$\xleftarrow{x}$	
Use factorization(N) to compute a square root of x modulo N, as in Section 1;		
	Zero knowledge interactive proof of Figure 3 that the prover knows a square root of x modulo N.	

Figure 5: Second Interactive Proof of Knowledge of Factorization

4 Proof for Knowledge of Factorization

We return finally to the zero knowledge interactive proof that your associate knows the prime factorization of some given integer N. This proof was discovered by Tompa and Woll [20], where details of the proof of correctness may be found.

From the result described in Section 1, it is sufficient for the prover to demonstrate the ability to extract a square root of arbitrary quadratic residues modulo N. Thus, a first attempt might look something like the protocol given in Figure 4. Like the zero knowledge interactive proof of Section 3, this one can be repeated with new random values t in order to increase the verifier's confidence that the prover knows the factorization of N. It is clear, though, that this protocol is not zero knowledge, as the prover reveals a square root s of x that the verifier might not have known. In fact, even a verifier not deviating from the protocol of Figure 4 may learn a proper factor of N.

This problem is addressed in the amended protocol of Figure 5. The basis for this is that the verifier need not receive a square root of x modulo N in order to be convinced that the prover knows N's factorization: it is sufficient if the verifier is convinced that the prover *knows* such a square root. This protocol is an improvement over that in Figure 4, but a moment's reflection shows that it is still not zero knowledge. By telling the verifier that it knows a square root of x modulo N, the prover is releasing the information that x is a quadratic residue modulo N, which a deviating verifier may not have known. (Recall from Section 1 that quadratic residuosity is not easy for the verifier to determine unaided.)

As in the protocol of Section 2, the prover should be unwilling to reveal the fact that x is a quadratic residue modulo N before being convinced that the verifier already knows that fact. The protocol of Figure 6 corrects this, and is indeed zero knowledge [20].

Acknowledgements

I am grateful to Mike Fischer and Heather Woll for numerous enlightening discussions about these topics.

Prover $(N,$ factorization$(N))$	*Comm. Channel*	*Verifier* (N)
		choose $t \in Z_N^*$ randomly and uniformly;
		$x \leftarrow t^2 \bmod N$;
	$\xleftarrow{x}$	
	Zero knowledge interactive proof of Figure 3 that the *verifier* knows a square root of x modulo N.	
Use factorization(N) to compute a square root of x modulo N, as in Section 1;		
	Zero knowledge interactive proof of Figure 3 that the *prover* knows a square root of x modulo N.	

Figure 6: Zero Knowledge Interactive Proof of Knowledge of Factorization

References

[1] L. M. Adleman, K. Manders, and G. Miller, "On Taking Roots in Finite Fields", *18th Annual Symposium on Foundations of Computer Science*, Providence, Rhode Island, October-November 1977, 175-178.

[2] L. M. Adleman, and K. S. McCurley, "Open Problems in Number Theoretic Complexity", *Discrete Algorithms and Complexity — Proceedings of the Japan-US Joint Seminar. Perspectives in Computing*, vol. 15, Academic Press, San Diego, 1987, 237-262.

[3] D. Angluin, "Lecture Notes on the Complexity of Some Problems in Number Theory", Technical Report 243, Yale University, August 1982.

[4] D. Angluin and D. Lichtenstein, "Provable Security of Cryptosystems: a Survey", Technical Report TR-288, Yale University, October 1983.

[5] E. Berlekamp, "Factoring Polynomials over Large Finite Fields", *Mathematics of Computation*, vol. 24, 1970, 713-735.

[6] M. Blum, "Three Applications of the Oblivious Transfer", University of California at Berkeley, unpublished manuscript, September 1981.

[7] G. Brassard and C. Crépeau, "Non-Transitive Transfer of Confidence: A Perfect Zero-Knowledge Interactive Protocol for SAT and Beyond", *27th Annual Symposium on Foundations of Computer Science*, Toronto, Ontario, October 1986, 188-195.

[8] D. Chaum, "Demonstrating that a Public Predicate can be Satisfied Without Revealing Any Information About How", *Advances in Cryptology — Crypto '86 Proceedings.* A. M. Odlyzko (ed.), *Lecture Notes in Computer Science*, vol. 263, Springer-Verlag, Berlin, 1987, 195-199.

[9] D. Chaum and J. van de Graaf, "An Improved Protocol for Demonstrating Possession of a Discrete Logarithm and Some Generalizations", *Eurocrypt 87*, Amsterdam, The Netherlands, April 1987, IV-15 to IV-21.

[10] U. Feige, A. Fiat, and A. Shamir, "Zero Knowledge Proofs of Identity", *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, New York, N.Y., May 1987, 210-217.

[11] M. J. Fischer, S. Micali, and C. Rackoff, "A Secure Protocol for the Oblivious Transfer", *Eurocrypt 84*.

[12] M. J. Fischer, S. Micali, C. Rackoff, M. Tompa, and D. K. Wittenberg, "An Oblivious Transfer Protocol", in preparation.

[13] Z. Galil, S. Haber, and M. Yung, "Mimimum-Knowledge Interactive Proofs for Decision Problems", 1987, to appear.

[14] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that Yield Nothing But their Validity and a Methodology of Cryptographic Protocol Design", *27th Annual Symposium on Foundations of Computer Science*, Toronto, Ontario, October 1986, 174-187.

[15] S. Goldwasser, S. Micali, and C. Rackoff, "The Knowledge Complexity of Interactive Proof-Systems", *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*, Providence, Rhode Island, May 1985, 291-304.

[16] J. D. Lipson, *Elements of Algebra and Algebraic Computing*, Addison-Wesley, Reading, Massachusetts, 1981.

[17] M. O. Rabin, "Digitalized Signatures and Public-Key Functions as Intractable as Factorization", Technical Report MIT/LCS/TR-212, M.I.T., January 1979.

[18] M. O. Rabin, "Probabilistic Algorithms in Finite Fields", *SIAM Journal on Computing*, vol. 9 (1980), 273-280.

[19] M. O. Rabin, "How to Exchange Secrets", unpublished manuscript, 1981.

[20] M. Tompa and H. Woll, "Random Self-Reducibility and Zero Knowledge Interactive Proofs of Possession of Information", *28th Annual Symposium on Foundations of Computer Science*, Los Angeles, California, October 1987, 472-482.

[21] H. Woll, "Reductions among Number Theoretic Problems", *Information and Computation*, vol. 72, no. 3 (March 1987), 167-179.