How To Prove Yourself: Practical Solutions to Identification and Signature Problems

Amos Fiat and Adi Shamir
Department of Applied Mathematics
The Weizmann Institute of Science
Rehovot 76100, Israel

Abstract.

In this paper we describe simple identification and signature schemes which enable any user to prove his identity and the authenticity of his messages to any other user without shared or public keys. The schemes are provably secure against any known or chosen message attack if factoring is difficult, and typical implementations require only 1% to 4% of the number of modular multiplications required by the RSA scheme. Due to their simplicity, security and speed, these schemes are ideally suited for microprocessor-based devices such as smart cards, personal computers, and remote control systems.

1. Introduction

Creating unforgeable ID cards based on the emerging technology of smart cards is an important problem with numerous commercial and military applications. The problem becomes particularly challenging when the two parties (the prover A and the verifier B) are adversaries, and we want to make it impossible for B to misrepresent himself as A even after he witnesses and verifies arbitrarily many proofs of identity generated by A. Typical applications include passports (which are often inspected and photocopied by hostile governments), credit cards (whose numbers can be copied to blank cards or used over the phone), computer passwords (which are vulnerable to hackers and wire tappers) and military command and control systems (whose terminals may fall into enemy hands). We distinguish between three levels of protection:

1) Authentication schemes: A can prove to B that he is A, but someone else cannot prove to B that he is A.

2) Identification schemes: A can prove to B that he is A, but B cannot prove to someone else that he is A.

3) Signature schemes: A can prove to B that he is A, but B cannot prove even to himself that he is A.

Authentication schemes are useful only against external threats when A and B cooperate. The distinction between identification and signature schemes is subtle, and manifests itself mainly when the proof is interactive and the verifier later wants to prove its existence to a judge: In identification schemes B can create a credible transcript of an imaginary communication by carefully choosing both the questions and the answers in the dialog, while in signature schemes only real communication with A could generate a credible transcript. However, in many commercial and military applications the main problem is to detect forgeries in real time and to deny the service,

A.M. Odlyzko (Ed.): Advances in Cryptology - CRYPTO '86, LNCS 263, pp. 186-194, 1987.

access or response that the forger wants. In these cases the transcript and judge are irrelevant, and the two types of schemes can be used interchangeably.

2. Interactive Identification

2.1 Background

The new identification scheme is a combination of zero-knowledge interactive proofs (Goldwasser, Micali and Rackoff [1985]) and identity-based schemes (Shamir [1984]). It is based on the difficulty of extracting modular square roots when the factorization of n is unknown. A related protocol for proving the quadratic residuosity of numbers was presented by Fischer Micali and Rackoff at Eurocrypt 84 (it did not appear in the proceedings), but the new protocol is faster and requires less communication. The main contribution of this paper is to show the relevance of such protocols to practical identification and signature problems.

The scheme assumes the existence of a trusted center (a government, a credit card company, a computer center, a military headquarters, etc.) which issues the smart cards to users after properly checking their physical identity. No further interaction with the center is required either to generate or to verify proofs of identity. An unlimited number of users can join the system without degrading its performance, and it is not even necessary to keep a list of all the valid users. Interaction with the smart cards will not enable verifiers to reproduce them, and even complete knowledge of the secret contents of all the cards issued by the center will not enable adversaries to create new identities or to modify existing identities. Since no information whatsoever is leaked during the interaction, the cards can last a lifetime regardless of how often they are used.

2.2 The Scheme

Before the center starts issuing cards, it chooses and makes public a modulus n and a pseudo random function f which maps arbitrary strings to the range $[0, n)$. The modulus n is the product of two secret primes p and q, but unlike the RSA scheme, only the center knows the factorization of the modulus and thus everyone can use the same n. The function f should be indistinguishable from a truly random function by any polynomially bounded computation. Goldreich Goldwasser and Micali [1984] describe a particular family of functions which is provably strong in this sense, but we believe that in practice one can use simpler and faster functions (e.g., multiple DES) without endangering the security of the scheme.

When an eligible user applies for a smart card, the center prepares a string I which contains all the relevant information about the user (his name, address, ID number, physical description, security clearance etc.) and about the card (expiration date, limitations on validity, etc). Since this is the information verified by the scheme, it is important to make it detailed and to double check its correctness. The center then performs the following steps:

1. Compute the values $v_j = f(I, j)$ for small values of j.
2. Pick k distinct values of j for which v_j is a quadratic residue $\pmod n$ and compute the smallest square root s_j of $v_j^{-1} \pmod n$.
3. Issue a smart card which contains I, the k s_j values, and their indices.

Remarks:

1. To simplify notation in the rest of this paper, we assume that the first k indices $j = 1, 2, \ldots, k$ are used.

2. For non-perfect functions f, it may be advisable to randomize I by concatenating it to a long random string R which is chosen by the center, stored in the card, and revealed along with I.
3. In typical implementations, k is between 1 and 18, but larger values of k can further reduce the time and communication complexities of the scheme.
4. n should be at least 512 bits long. Factoring such moduli seems to be beyond reach with today's computers and algorithms, with adequate margins of safety against foreseeable developments.
5. The center can be eliminated if each user chooses his own n and publishes it in a public key directory. However, this RSA-like variant makes the schemes considerably less convenient.

The verification devices are identical standalone devices which contain a microprocessor, a small memory, and I/O interface. The only information stored in them are the universal modulus n and function f. When a smart card is inserted into a verifier, it proves that it knows $s_1, \ldots, s_k$ without giving away any information about their values. The proof is based on the following protocol:

1. A sends I to B.
2. B generates $v_j = f(I, j)$ for $j = 1, \ldots, k$.

Repeat steps 3 to 6 for $i = 1, \ldots, t$:

3. A picks a random $r_i \in [0, n)$ and sends $x_i = r_i^2 \pmod{n}$ to B.
4. B sends a random binary vector $(e_{i1}, \ldots, e_{ik})$ to A.
5. A sends to B:

$$y_i = r_i \prod_{e_{ij}=1} s_j \pmod{n}.$$

6. B checks that

$$x_i = y_i^2 \prod_{e_{ij}=1} v_j \pmod{n}.$$

Remarks:

1. The verifier B accepts A's proof of identity only if all the t checks are successful.
2. To decrease the number of communicated bits, A can hash x_i by sending B only the first 128 bits of $f(x_i)$ in step 3. B can check the correctness of this value in step 6 by applying f to the right hand side of the equation and comparing the first 128 bits of the results.
3. A can authenticate a particular message m (e.g., an instruction to a remote control system or a program sent to a remote computer) without having to extract new square roots by sending B the first 128 bits of $f(m, x_i)$ in step 3. If B knows m, he can easily check this value in step 6. A is fully protected against modifications and forgeries of his messages by the pseudo random nature of f, but this is not a real signature scheme: without participating in the interaction, a judge cannot later decide if a message is authentic.

2.3 Security

Lemma 1: If A and B follow the protocol, B always accepts the proof as valid.

Proof: By definition

$$y_i^2 \prod_{e_{ij}=1} v_j = r_i^2 \prod_{e_{ij}=1} (s_j^2 v_j) = r_i^2 = x_i \pmod{n}.$$

□

Lemma 2: Assume that A does not know the s_j and cannot compute in polynomial time the square root of any product of the form $\prod_{j=1}^{k} v_j^{c_j} \pmod{n}$ ($c_j = -1, 0$ or $+1$, not all of them zero). If B follows the protocol (and A performs arbitrary polynomial time computations), B will accept the proof as valid with probability bounded by 2^{-kt}.

Proof (Sketch): A can cheat by guessing the correct e_{ij} vectors and sending

$$x_i = r_i^2 \prod_{e_{ij}=1} v_j \pmod{n} \quad \text{and} \quad y_i = r_i.$$

However, the probability of this event is only 2^{-k} per iteration and 2^{-kt} for the whole protocol. To increase this probability, A must choose the x_i values in such a way that for a non-negligible fraction of them he can compute the square roots y_i' and y_i'' of

$$x_i \Big/ \prod_{e_{ij}=1} v_j \pmod{n}$$

for two vectors e_{ij}' and e_{ij}''. The ratio $y_i'/y_i'' \pmod{n}$ is of the form $\prod_{j=1}^{k} s_j^{c_j} \pmod{n}$. This contradicts the assumption, since A himself can simulate B's random questions and thus compute in expected polynomial time a value we assumed he cannot compute. □

Lemma 3: For a fixed k and arbitrary t, this is a zero-knowledge proof.

Proof (Sketch): The intuitive (but non-rigorous) reason the proof reveals no information whatsoever about the s_j is that the x_i are random squares, and each y_i contains an independent random variable which masks the values of the s_j. All the messages sent from A to B are thus random numbers with uniform probability distributions, and cheating by B cannot change this fact.

To prove this claim formally, in the full paper we exhibit a probabilistic algorithm which simulates the communication between A and B without knowing the s_j with a probability distribution which is indistinguishable from the real distribution. The expected running time of this algorithm is $t \cdot 2^k$ times the sum of the expected running times of A and B. By assumption, this running time is polynomial. □

Remarks:

1. Throughout this paper, 2^{kt} is assumed to be much smaller than the time required to factor the modulus n.
2. The quadratic residuosity protocol of Fischer Micali and Rackoff is a special case of this protocol with $k = 1$. The main practical advantage of the new protocol is that for the same security we can use only the square root of the number of iterations, which reduces the time and communication complexities of the protocol and its applications.
3. An adversary who records polynomially many proofs of identity cannot increase his chance of success: If he reuses a recorded x_i, he can playback the recorded answers only if the questions happen to be the same. Since A uses each x_i only once, the probability of

success is still 2^{-kt}.

4. In the parallel version of this protocol, A sends all the x_i, then B sends all the e_{ij}, and finally A sends all the y_i. This version is not zero-knowledge for technical reasons, but its security can be formally proven by the techniques developed in Section 3.

The 2^{-kt} probability of forgery is an absolute constant, and thus there is no need to pick large values of k and t as a safeguard against future technological developments. In most applications, a security level of 2^{-20} suffices to deter cheaters. No one will present a forged passport at an airport, give a forged driver's license to a policeman, use a forged ID badge to enter a restricted area, or use a forged credit card at a department store, if he knows that his probability of success is only one in a million. In all these applications, the forged ID card (rather than the transcript of the communication) can be presented to a judge as evidence in a trial. Even if the only penalty for a failed attempt is the confiscation of the card, and smart cards cost only \$1 to manufacture, each success will cost about one million dollars. For national security applications, we can change the security level to 2^{-30}: Even a patient adversary with an unlimited budget, who tries to misrepresent himself 1000 times each day, is expected to succeed only once every 3000 years.

2.4 Complexity

To attain a 2^{-20} level of security, it suffices to choose $k = 5$, $t = 4$ (for 2^{-30}, increase these values by 1). The average number of modular multiplications required to generate or verify a proof of identity in this case is $t(k+2)/2 = 14$. The number of bytes exchanged by the parties during the proof is 323, and the secret s_j values can be stored in a 320 byte ROM. Even better performance can be obtained by increasing k to 18 (a 1152 byte ROM). If we use e_{ij} vectors with at most three 1's in them, we have a choice of 988 possible vectors in each iteration. With $t = 2$ iterations, the security level remains about one in a million, but the number of transmitted bytes drops to 165 and the average number of modular multiplications drops to 7.6 (which is two orders of magnitude faster than the 768 multiplications required by the RSA scheme). Note that the 2×18 e_{ij} matrix is so sparse that B has to generate at most 6 out of the 18 v_j values to verify the proof.

The time, space, communication and security of the scheme can be traded off in many possible ways, and the optimal choices of k, t and the e_{ij} matrix depends on the relative costs of these resources. Further improvements in speed can be obtained by parallelizing the operations. Unlike the RSA scheme, the two parties can pipeline their operations (with A preparing x_{i+1} and y_{i+1} while B is still checking x_i and y_i), and use parallel multipliers to compute the product of v_j or s_j values in $\log k$ depth. Since the protocol uses only multiplication (and no gcd or division operations which are hard to parallelize), each iteration of the protocol is in NC, and thus the scheme is suitable for very high speed applications.

3. Signatures

3.1 The Scheme

B's role in the interactive identification scheme is passive but crucial: The random e_{ij} matrix he sends contains no information but its unpredictability prevents cheating by A. To turn this identification scheme into a signature scheme, we replace B's role by the function f and obtain the following protocol:

To sign a message m:

1. A picks random $r_1, \ldots, r_t \in [0, n)$ and computes $x_i = r_i^2 \pmod{n}$.
2. A computes $f(m, x_1, \ldots x_t)$ and uses its first kt bits as e_{ij} values $(1 \leq i \leq t,\ 1 \leq j \leq k)$.
3. A computes

$$y_i = r_i \prod_{e_{ij}=1} s_j \pmod{n} \qquad \text{for} \quad i = 1, \ldots, t$$

and sends I, m, the e_{ij} matrix and all the y_i to B.

To verify A's signature on m:

1. B computes $v_j = f(I, j)$ for $j = 1, \ldots, k$.
2. B computes

$$z_i = y_i^2 \prod_{e_{ij}=1} v_j \pmod{n} \qquad \text{for} \quad i = 1, \ldots, t.$$

3. B verifies that the first kt bits of $f(m, z_1, \ldots, z_t)$ are e_{ij}.

3.2 Security

The formal proof of security in this extended abstract assumes that n is sufficiently large and that f is a truly random function. Consequently, there can be no generic attack which breaks the scheme for any n and f unless factoring is easy. Practical implementations which use particular moduli n_0 and psuedo-random functions f_0 may still be vulnerable to specialized attacks, but they mearly show that n_0 is too small or that f_0 is demonstrably non-random. When n_0 is at least 512 bits long and f_0 is sufficiently strong (*e.g.*, multiple DES with a fixed cleartext and variable key), such attacks are quite unlikely.

Lemma 4: If A and B follow their protocols, B always accepts the signature as valid.

Proof: By definition,

$$z_i = y_i^2 \prod_{e_{ij}=1} v_j = r_i^2 \prod_{e_{ij}=1} (s_j^2 v_j) = r_i^2 = x_i \pmod{n},$$

and thus $f(m, z_1, \ldots, z_t) = f(m, x_1, \ldots, x_t)$. □

Lemma 5: A chooses a particular signature among all the possible signatures for the message m with uniform probability distribution.

Proof: Given a signature (e_{ij} matrix and y_i values), it is possible to recreate $r_1^2, \ldots, r_k^2 \pmod{m}$ uniquely, and $r_1, \ldots, r_k$ in exactly 4^k ways. Since A chooses the r_i at random, the various signatures are chosen with equal probabilities. □

Lemma 6: Let AL be any polynomial time probabilistic algorithm which accepts $n, v_1, \ldots, v_k$ and the signatures of arbitrary messages $m_1, m_2, \ldots$ of its choice, and produces a valid signature of another message m_0 of its choice. If the complexity of factoring and 2^{kt} grow non-polynomially with the size of n, AL cannot succeed with non-negligible probability for random functions f.

Proof (Sketch): By contradiction. Using a simple combinatorial argument, we can prove that a polynomial time variant AL' of AL can compute a square root of some product $\prod_{j=1}^{k} v_j^{c_j}$

(mod n) ($c_j = -1$, 0, or $+1$, not all of them zero) with a similar probability of success.

To turn AL' into a factoring algorithm for n, pick random $s_1, \ldots, s_k$ and define $v_j = s_j^2 \pmod n$. Execute AL' with $n, v_1, \ldots, v_k$ as input, and use the s_j to supply the signatures of $m_1, m_2, \ldots$ requested by AL'. The output of AL' is a square root Q of $\prod_{j=1}^{k} v_j^{c_j} \pmod n$, but another square root S ($= \prod_{j=1}^{k} s_j^{c_j} \pmod n$) is already known. By Lemma 5, AL' cannot find out which one of the four possible roots is S by analysing the given signatures of $m_1, m_2, \ldots$. Consequently, $\gcd(Q - S, n)$ is a proper factor of n with probability 1/2. By repeating this procedure several times, we can make this probability arbitrarily close to 1. □

It is easy to forge signatures for arbitrary messages m_0 in time T with probability $T \cdot 2^{-kt}$ by guessing the e_{ij} matrix T times. A refinement of Lemma 6 shows that when the complexity of factoring is considerably higher than 2^{kt}, this attack is essentially optimal:

Lemma 7: Let AL be any probabilistic algorithm of the type described in Lemma 6. If AL runs in time T and succeeds with probability $(1+\epsilon)T2^{-kt}$ for random functions f, then n can be factored with non negligible probability in time $T^2 \cdot 2^{kt}$.

Proof: Will be given in the full paper. □

Corollary 8: If k and t are chosen so that the ratio between the complexity of factoring and 2^{kt} grows non-polynomially with the size of n, then the $T2^{-kt}$ probability of forgery is tight for polynomial time attacks.

Discussion

The sequential version of the interactive identification scheme is zero-knowledge and thus B cannot deduce any information whatsoever about the s_j from his interaction with A. The parallel identification scheme and the signature scheme, on the other hand, cannot be proven zero-knowledge for very subtle technical reasons. In fact, strong signature schemes cannot be zero-knowledge by definition: If everyone can recognize valid signatures but no one can forge them, B cannot generate by himself A's messages with the same probability distribution. However, corollary 8 shows that the information about the s_j's that B gets from signatures generated by A is so implicit that it cannot be used to forge new signatures, and thus the signature scheme is provably secure (if factoring is difficult) even though it is not zero-knowledge.

3.3 Complexity

In the proposed signature scheme, an adversary knows in advance whether his signature will be accepted as valid, and thus by experimenting with 2^{kt} random r_i values, he is likely to find a signature he can send to B. Consequently, the product kt must be increased from 20 to at least 72 when we replace the identification scheme by a signature scheme.

A choice of $k = 9$, $t = 8$ attains the desired 2^{-72} security level. The private key can be stored in a 576 byte ROM, and each signature requires 521 bytes. The average number of modular multiplications for this choice is $t(k+2)/2 = 44$.

By doubling the key size to 1152 bytes ($k = 18$), we can reduce the size of each signature to 265 bytes ($t = 4$) without changing the 2^{-72} security level. By optimizing the order of the multiplications to compute the t subset products simultaneously, we can reduce their average number to 32. This is only 4% of the number of multiplications required in the RSA signature scheme. Other points along the tradeoff curve for the 2^{-72} security level are summarized in Table 1.

Table 1: Tradeoffs for k and t at the 2^{-72} Security Level

k	t	Secret Key Size (in bytes)	Signature Size (in bytes)	Average # Mult. (Standard)	Average # Mult. (Optimized)	Average # v_i's B generates
1	72	64	4608 + 9	108	108	1
2	36	128	2304 + 9	72	64	2
3	24	192	1536 + 9	60	49	3
4	18	256	1152 + 9	54	46	4
6	12	384	768 + 9	48	41	6
8	9	512	576 + 9	45	45	8
9	8	576	512 + 9	44	44	9
12	6	768	384 + 9	42	35	12
18	4	1152	256 + 9	40	32	17
24	3	1536	192 + 9	39	28	21
36	2	2304	128 + 9	38	30	24
72	1	4608	64 + 9	37	37	36

4. Extensions

A unique feature of the new identification and signature schemes is that it is possible to change their level of security after the key has been chosen. Consider, for example, an access card with $k = 18$ s_j values: The fast screening procedure at the entrance to the building will be controlled with $t = 1$ (2^{-18} security level), access to the computer room will be controlled by $t = 2$ (2^{-36} security level), while any usage of the computer will leave signed audit trails with $t = 4$ (2^{-72} security level). The only dangerous case is the simultaneous usage of the same s_j values in a parallel identification scheme with a large t and in a signature scheme with a small t (an unlikely combination), which is susceptible to an active playback attack.

Since the verification devices store only small amounts of publicly available information, it is possible to standardize them: One device can store several values of n and f and thus check a variety of personal, financial and occupational ID cards provided by many independent organizations. This possibility is particularly important in department stores which have to recognize many types of credit cards or in check cashing situations which require three ID cards of many possible types.

The proposed schemes can be generalized in a variety of ways. For example, the square roots can be replaced by cubic or higher roots, the e_{ij} matrix can be made non-binary, and the usage of r_i and s_j values can be made more symmetric in the generation of each y_i value. A more radical generalization is suggested by Goldreich, Micali and Wigderson's recent discovery of zero knowledge proofs for NP problems: It is now possible to use any instance of any NP complete problem as the basis for identification and signature schemes. Shamir later improved the time and communication complexities of these proofs, but their practical significance is still unclear.

5. Acknowledgements

We would like to thank Mike Fischer, Oded Goldreich, Shafi Goldwasser, Silvio Micali, Charlie Rackoff, Claus Schnorr and Avi Wigderson for inspiring many of the ideas presented in this paper.

6. Bibliography

1. Fischer, Micali and Rackoff [1984]: *A Secure Protocol for the Oblivious Transfer*, presented at Eurocrypt, April 1984.
2. Goldreich, Goldwasser and Micali [1984]: *How to Construct Random Functions*, 25*th* Symposium on Foundations of Computer Science, October 1984.
3. Goldreich, Micali and Wigderson [1986]: *Proofs that Yield Nothing But the Validity of the Assertion and the Methodology of Cryptographic Protocol Design*, submitted to 27*th* Symposium on Foundations of Computer Science, November 1986.
4. Goldwasser, Micali and Rackoff [1985]: *The Knowledge Complexity of Interactive Proof Systems*, 17*th* ACM Symposium on Theory of Computation, May 1985.
5. Shamir [1984]: *Identity-Based Cryptosystems and Signature Schemes*, Proceedings of Crypto '84, Lecture Notes in Computer Science no. 196, Springer Verlag 1985.