Strong Signature Schemes

by

Shafi Goldwasser* — Lab. for Computer Science, MIT

Silvio Micali* — Computer Science Department, University of Toronto

Andy Yao — IBM Research, San Jose

Abstract

The notion of digital signature based on trapdoor functions has been introduced by Diffie and Hellman[3]. Rivest, Shamir and Adleman[8] gave the first number theoretic implementation of a signature scheme based on a trapdoor function. If f is a trapdoor function and m a message, $f^{-1}(m)$ is the signature of m. The signature can be verified by computing $f(f^{-1}(m))=m$. This approach presents the following problems even when f is hard to invert:

1) there may be special message spaces (or subsets of them) that are easy to sign without knowing the *trapdoor* information
2) it is possible to forge the signature of random numbers; this violates the requirements of many protocols
3) given a polynomial number of signed messages, it may be possible to sign a new one without knowing the trapdoor information.

We solve the above problems by exhibiting two signature schemes for which any strategy of an adversary, who has seen all previously signed messages, that has a moderate success in forging even a single additional signature, is transformable to a fast algorithm for factoring or inverting the RSA function. This provably holds for all message spaces with all possible Probability distributions. Thus, in particular, given the signature of m, forging the signature of $m+1$ or 2m or $2^s m$ is as hard as factoring. The two signature schemes

* Work was done when the first two authors were at the University of California at Berkeley, and supported by NSF grant MCS 82-04506. In addition, Shafi Goldwasser was supported by a Bantrell Fellowship.

1. Introduction

The design of cryptographic protocols using trapdoor and one-way functions has received considerable attention in the past few vears [1-8]. More recently, attention has been paid to provide rigorous correctness proofs based on simple mathematical assumptions, for example, in coin flipping (Blum [1]), mental poker (Goldwasser and Micali [4]). It is perhaps reasonable to speculate at this time that all cryptographic protocols can eventually be designed to be provably secure under simple assumptions, such as factoring large numbers or inverting RSA functions are computationally intractable in the appropriate sense.

In this paper, we will study in the above light the basic problem of *digital signature*, which was one of the original applications for trapdoor functions in Diffie and Hellman [3]. It is also important as the ability to sign messages is often a prerequisite in other cryptographic protocols.

Diffie and Hellman [3] proposed the following solution to the signature problem. Let M be the *message space.* Every user A in the public network has an encryption algorithm E_A (as specified by a *public key* K_A) listed in a secure public file, and keeps secret a decryption algorithm D_A (specified by a *private key* K_A'), where $E_A(D_A(m))=m$ for all $m \in M$. The function E_A is a *trapdoor function*, in the sense that it is computationally hard to compute $D_A(m)$ without knowing the secret key K_A'. For user A to send the signature of a message $m \in M$ to another user B, A will compute $D_A(m)$, and send both m and $D_A(m)$ to B. Since nobody else is likely to be able to produce $D_A(m)$, B can compute $E_A(D_A(m))=m$ to verify that he has received a properly signed message from A and retains $D_A(m)$ as a proof that A has signed m.

Several implementations of the above approach have been suggested in the literature [6][8]. For example, in the Rabin signature scheme [6], user A enters in the public file a large composite integer N, which is the product of two large primes, while keeping secret the prime factors as the private key. Let $E_A(m)=m^2$ mod N. To sign a message m, A computes $\sqrt{m}$ mod N which is easy to do knowing the prime factors of N. On the other hand, computing square roots mod N without knowing the prime factors is very difficult, and has been shown by Rabin [6] to be equivalent to factoring N.

Although the above general approach is intuitively attractive, it is not easy to give a precise theoretical foundation for its security. In fact, there are several potential difficulties. Firstly, it cannot be used for a very dense message space; for instance, anyone can easily produce in the Rabin scheme a valid signed message, simply by trying to square different x's in Z_N^* until an x^2 mod $N \in M$ is found. Then, x is a valid signature of x^2 mod N. Secondly, if the message space is very sparse, it might happen that the function $D_A(m)$ will be easy to compute for $m \in M$; for example, it is conceivable that taking the square root modulo N of m of a special form is easy, even though the general problem is as hard as factoring integers. Finally, there is also the possibility that forging new messages may become easy if an adversary has available some previously signed messages. This problem can be illustrated with a simple example. Let $y = D_A(m) = \sqrt{m}$ mod N in the Rabin scheme. Then, $4m$ is easy to sign. Simply let, $D_A(4m) = 2y$ mod N.

There are modifications that can alleviate some of the above problems. For example, by signing m with $D_A(mu_A)$, where u_A we a special string, it lessens the dense-message-space problem. Still, there has not been any published signature scheme based on the public-key idea that has been proved to be secure under simple mathematical assumptions such as integer factorization.

In this paper we will demonstrate this possibility by presenting two signature schemes. We prove that for any message space M and any non-negligible fraction $\varepsilon > 0$, the probability that an adversary, who has seen a polynomial number of signed messages, can sign even a single additional message $m \in M$ not previously signed, is less than ε. The security of the first scheme can be proved under the intractability assumption of factoring composite numbers which are products of large primes. The security of the second scheme can be proved under the intractability assumption of inverting the RSA[9] function.

We remark that there are other types of signature schemes in the literature, such as the Rabin's scheme [7], and Lamport-Diffie scheme as given in [3]. This latter scheme is of special interest, as our signature scheme in Section 4 can be viewed as an improvement of their scheme. The main drawback in [3] is the huge amount of information that has to be put in the public file. A rough estimate shows that in order to sign messages for a total length of n bits, a user A should publicize at least $2n$ bits.

2. Strong Signature Schemes

As our purpose here is to present the new signature method, and to prove the security of this particular scheme, we will only informally describe what a *strong signature scheme* (SSS) is. An SSS has a parameter n, which can be thought of roughly as the number of bits in the argument of the trapdoor functions employed. We are interested in constructing schemes which become computationally impossible to break when n becomes large. From now on, an efficient procedure will mean a probabilistic polynomial(in n) -time algorithm.

Let b be a fixed, positive, integer. Assume that each user will sign at most n^b bits of messages. A *signature scheme S* specifies, given n, how a user A generates efficiently a pair of binary strings (P_A,S_A), where P_A is the *public information* to be listed in a public file and S_A is the *private information* to be kept secret. $|P_A|$, the number of bits in the description of P_A, must be "negligible" with respect to n^b. In the two following implementations of an SSS, $|P_A|$ = O(n). At any time A is capable of producing efficiently from n and S_A a *signed message m'* for any message $m \in M$. It should be easy for any one to compute m from m' and P_A, and verify that the latter is a legitimate signed message of m.

The security requirement we wish to impose is that an adversary cannot hope to forge a new signature, even after seeing a large number of genuine signed messages from A. Let $T=m_1,m_2,...,m_h$ be a sequence of messages from M with at most n^b bits, and let $T'=m'_1,m'_2,...,m'_h$ denote a sequence of corresponding signed messages from A. A ***forger B*** is an efficient algorithm which takes P_A,n and a sequence of signed messages $m'_1,m'_2,...,m'_h$ as inputs, and outputs a message m and a string y. We say that A is *secure* if, for every forger B, and all T, the probability of producing a valid signature $y=m'$ of message m, with $m \neq m_i$ for all i is $o(1)$ as n goes to infinity.

We now proceed to introduce two number theoretic implementations of a strong signature scheme. The first scheme's security solely depends on the intractability of factoring. The second one is based on the intractability of inverting the RSA function(for the exact assumption, see section 4.1).

Though the intractability of factoring is a weaker assumption than the intractability of inverting the RSA function, our second scheme may be more attractive in practice.

3. A Strong Signature Scheme Based on Factoring

In this section we present an SSS whose security depends on the intractability of factoring. Messages will be signed in a bit-by-bit fashion. The symbol $|x|$ will denote the number of bits in the binary representation of integer x. The symbol (x,y) will denote the greatest common divisor of integers x and y. $Z_N^* = \{x < N \mid (x,N) = 1\}$.

3.1 The Factoring Assumption

Let us define H, the set of hard integers. $H = \{N \mid N = pq$ where p, q are primes such that $|p| = |q|$ or $|p| = |q| + 1\}$. Randomly selected elements of H are the hardest inputs for all known factoring algorithms. H_k will denote the set of hard integers of length k.

In this section we make the following assumption about the intractability of factoring:

The Factoring Assumption: Let $0 < \varepsilon < 1$. For each positive integer k, let $C_{k,\varepsilon}$ be the minimum size of circuits C that factor n for a fraction ε of the $n \in H_k$. Then, for every $0 < \varepsilon < 1$ and every polynomial Q, for all sufficiently large k: $C_{\varepsilon,k} > Q(k)$.

3.2 Number Theoretic Background

Let p be a prime. For any integer a, the *Legendre symbol* of $a \in Z_p^*$, $(\frac{a}{p})$ is defined as: $(\frac{a}{p}) = a^{\frac{p-1}{2}}$ mod p. The equation $a \equiv x^2$ mod p has solutions if and only if $(\frac{a}{p}) = 1$.

A *quadratic residue mod N* is an element $a \in Z_N^*$ such that the equation $a \equiv x^2$ mod N has solutions in Z_N^*. This is the case if and only if $a \equiv x^2$ mod p is solvable for every prime p dividing N.

Let N be the product of two distinct primes, N = pq, then every quadratic residue, a, mod N has 4 square roots mod N: x, $-x$ *mod* N, y, $-y$ *mod* N. We recall that if $x \in Z_N^*$, -x mod N = N - x. The *Jacobi symbol* is a function from the integers into $\{0,1,-1\}$. For $x \in Z_N^*$, the Jacobi symbol of x is denoted by $(\frac{x}{N})$, where $(\frac{x}{N}) = (\frac{x}{p}) \cdot (\frac{x}{q})$. However, $(\frac{x}{N})$ can be computed in $O(|N|^3)$ {Time even when the prime factorization of N is not known.

In this section we will focus our attention on hard integers $N = pq$ where p and q are large prime factors such that $p \equiv q \equiv 3$ mod 4. The cryptographic relevance of such composite integers has been brought out by Blum [1].

Definition (Blum-integer): We will call a *Blum-integer* a hard integer N, N=pq, such that $p \equiv q \equiv 3$ mod 4. We let

$BI = \{N \in H \mid N = pq, p \equiv q \equiv 3 \ mod \ 4\}$.

Blum-integers have been successfully used in protocols for coin flipping [1], playing mental poker [4], and in procedures for generating pseudo random bit sequences [5] [6] [11].

The usefulness of these numbers derives from the following two facts which are proved in [1]:

Fact 1: For all $x \in Z_N^*$, $(\frac{x}{N}) = (\frac{-x}{N})$

Fact 2: For all $x,y \in Z_N^*$, if $x^2 \equiv y^2$ mod N and $x \not\equiv \pm y$ mod N then $(\frac{x}{N}) = -(\frac{y}{N})$

We now show a new application of these integers to digital signatures. Throughout this section, N will denote a Blum-integer. Let

$$Q_1^N = \{x \in Z_N^* \mid (\tfrac{x}{p}) = 1 \text{ and } (\tfrac{x}{q}) = 1\},$$
$$Q_2^N = \{x \in Z_N^* \mid (\tfrac{x}{p}) = -1 \text{ and } (\tfrac{x}{q}) = -1\},$$
$$Q_3^N = \{x \in Z_N^* \mid (\tfrac{x}{p}) = 1 \text{ and } (\tfrac{x}{q}) = -1\},$$
$$Q_4^N = \{x \in Z_N^* \mid (\tfrac{x}{p}) = -1 \text{ and } (\tfrac{x}{q}) = 1\}.$$

$Q_1^N \cdot Q_2^N \cdot Q_3^N$ and Q_4^N partition Z_N^* in equinumerous classes. Q_1 is the set of quadratic residues mod N. Let y_i be a given element in Q_i for $1 \le i \le 4$. Then,

Lemma 3 : For all $x \in Z_N^*$, there exists a unique $1 \le i \le 4$ such that xy_i mod $N \in Q_1$

proof: Let a and b be generators for Z_p^* and Z_q^* respectively, and $a \equiv 1$ mod q, $b \equiv 1$ mod p. By the Chinese Remainder Theorem such generators exist. expressed as $a^l b^j$ mod N for some $1 \le l < p$ and $1 \le j < q$. In addition, $x \in Q_1$ if and only if l and j are both even, $x \in Q_2$ if and only if l and j are both odd, $x \in Q_3$ if and only if l is even and j is odd, and finally $x \in Q_4$ if and only if l is odd and j is even. Let

$y_i = a^{l_i} b^{j_i}$ where $1 \le l_i < p$ and $1 \le j_i < q$. It follows immediately that for any $x \in Q_i$ such that $x = a^v b^u$ mod N, $1 \le u < p$ and $1 \le v < q$, $xy_i = a^{(v+l_i)} b^{(u+j_i)}$ mod N and $v + l_i$ mod $\varphi(N) \equiv 0$ mod 2, $u + j_i$ mod $\varphi(N) \equiv 0$ mod 2. Thus $xy_i \in Q_1$ for all $1 \le i \le 4$. Similarly if $x \in Q_i$ then $xy_j \notin Q_1$ for $j \neq i$. QED

Lemma 4: Let y in Q_3. Then $-y$ is in Q_4, and for all $x \in Q_1$, either $x \cdot y^{-1}$ mod $N \le \frac{N}{2}$ or $x \cdot (-y)^{-1}$ mod $N \le \frac{N}{2}$, but not both.

Proof: As $(\frac{y}{p}) = 1$ and $p \equiv 3$ mod 4, $(\frac{-y}{p}) = (\frac{-1}{p})(\frac{y}{p}) = ((-1)^{\frac{p-1}{2}} \; mod \; p)(\frac{y}{p}) = ((-1)^{1+2d} \; mod \; p)(\frac{y}{p}) = -(\frac{y}{p}) = -1$. Similarly, as $(\frac{y}{q}) = -1$ and $q \equiv 3$ mod 4, $(\frac{-y}{q}) = (\frac{-1}{q})(\frac{y}{q}) = -(\frac{y}{q}) = 1$. Thus, $-y \in Q_4$. Let $z = xy^{-1}$ mod N and $z' = x(-y)^{-1}$ mod N. Then, $zy = z'y$ and $z \le \frac{N}{2}$ if and only if $z' > \frac{N}{2}$.

Lemma 5: Let x be a quadratic residue mod N. Then, x has 4 square roots: x_1, x_2, x_3 and x_4 such that $x_1 \in Q_1$, $x_2 \in Q_2$, $x_3 \in Q_3$ and $x_4 \in Q_4$.
Proof: By fact 2 and lemma 4.

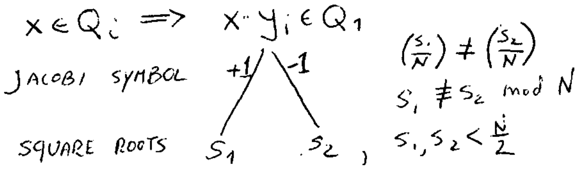
Fact 6: Given $x, y \in Z_N^*$ such that $x^2 \equiv y^2$ mod N and $(\frac{x}{N}) \neq (\frac{y}{N})$ mod N, the $gcd(x \pm y, N) = p$ or q.

Theorem(Rabin): Let $0 < \varepsilon < 1$ and N the product of two distinct primes. If there exists a polynomial time algorithm that on input N and quadratic residue $z \in Z_N^*$ computes a square root of z mod N for a fraction ε of the z's, then there exists a probabilistic polynomial$(|N|, \varepsilon^{-1})$ time algorithm for factoring N.

Idea of the proof: Pick x at random in Z_N^* and compute x^2 mod N. Use your polynomial time algorithm to try to compute a square root, y, of x^2 mod N. In case of success the square root computed, y, will not be x or $-x$ mod N, with probability $\frac{1}{2}$. By Lemma 6, if $y \not\equiv \pm x$, then you can factor N.

3.3 The Signature Primitives: f_0^N and f_1^N

Let $N = pq$ be a Blum-integer. Choose y at random from Q_3. Let $y_1 = 1$, $y_2 = -1$, $y_3 = $ y, $y_4 = -$y, and $R = \{y_i \mid i = 1,..,4\}$. Such an R will be called a *set of representatives* for Z_N^*.
Let x in Q_1^N. By lemma 3 and fact 4, xy_i is a quadratic residue mod N which has exactly two square roots mod N that are less than $\frac{N}{2}$. By facts 1 and 2, one of these square roots has Jacobi symbol +1, and the other has Jacobi symbol −1. See figure 1.


$x \in Q_i \Longrightarrow x \cdot y_i \in Q_1$
JACOBI SYMBOL
+1
-1
SQUARE ROOTS
s_1
s_2
$(\frac{s_1}{N}) \neq (\frac{s_2}{N})$
$s_1 \not\equiv s_2$ mod N
$s_1, s_2 < \frac{N}{2}$

Definition (f_0^N and f_1^N): The function $f_0^N : Q_1^N \to Z_N^*$ is defined as follows. Let $x \in Q_1^N$, let z be the unique square root of x mod N such that $(\frac{z}{N}) = 1$ and z $< \frac{N}{2}$. Then $f_0^N(x) =$ z. $f_1^N : Q_1^N \to Z_N^*$ is similarly defined: $f_1^N(x) =$ z iff $(\frac{z}{N}) = -1$, z $< \frac{N}{2}$ and $z^2 \equiv x$ mod N.

Lemma 7: : f_0^N and f_1^N are 1-to-1 functions.

Proof: By facts 1 and 2 and lemma 3.

Definition ($f_0^{N,R}$ and $f_1^{N,R}$): Let $R = \{y_1, y_2, y_3, y_4\}$ be a set of representatives for Z_N^*. Then $f_0^{N,R}$, $f_1^{N,R}$: $Z_N^* \to Z_N^*$ are defined as follows. For $1 \le i \le 4$, let $x \in Q_i^N$. Then, $f_0^{N,R}(x) = f_0^N(xy_i)$ and $f_1^{N,R}(x) = f_1^N(xy_i)$.

Note that $f_0^{N,R}$ and $f_1^{N,R}$ are 4-to-1 maps.

3.4 The Signature Function σ

On input N (Blum integer), R (set of representative for Z_N^*), x_0 (element in Z_N^*) and m (binary string), we define the algorithm $\sigma : \{0,1\}^* \to Z_N^*$ recursively: ($\cdot$ denotes concatenation)

$$\sigma(x_0, N, R, 0) = f_0^{N,R}(x_0)$$

$$\sigma(x_0, N, R, 1) = f_1^{N,R}(x_0)$$

$$\sigma(x_0, N, R, 0 \cdot m) = \sigma(f_0^{N,R}(x_0), N, m))$$

$$\sigma(x_0, N, R, 1 \cdot m) = \sigma(f_1^{N,R}(x_0), N, m))$$

Algo. A computes $\sigma(x_0, N, R, m)$.

```
Algo A:
 input  : $N$, $R$, $x_0$, $m = (b_1, \ldots, b_n)$
 output : T := $\sigma$(x$_0$,N,R,m)

 for j := 1 to n do
     if b$_j$ =0 then T := f$_0^{N,R}$(x$_0$)
     else T := f$_1^{N,R}$(x$_0$)
     $x_0 := T$
     end .
output T
```

For example, $\sigma(x_0, N, R, 010) := f_0^{N,R}(f_1^{N,R}(f_0^{N,R}(x_0)))$.

Definition : Given an $N \in BI$, a set of representatives for Z_N^* , $R = \{1, -1, y, -y \mid y \in Q_3^N\}$, an element $x_0 \in Z_N^*$, and a binary message m, $\sigma(x_0,N,R,m)$ is an (x_0,N,R)*-signature* of m.

Comment: We will sometimes call x_0 the starting location of an (x_0,N,R)*-signature* of m.

Lemma 8 : Given an $N = pq \in BI$, p, q, a set R of representative for Z_N^*, $x_0 \in Z_N^*$, and $m = b_1 \cdots b_k$, the $(x_0$,N,R)-signature of m can be computed in probabilistic O(k $|N|^3$) time.

Proof : Computing square roots mod N, $N = pq$, can be done in probabilistic O($|N|^3$) time if p and q are known. In fact, given a quadratic residue x mod N, the square roots of x mod p and q can be computed, respectively, in probabilistic O($|p|^3$) and O($|q|^3$) time (as, for example, in [1]). These square roots can then be combined through the Chinese Remainder Theorem to get the square roots of x mod N.

The next two claims show that given a signature x, and a message m, it is easy to verify that x indeed equals $\sigma(x_0,N,R,m)$.

Claim 9: Given a binary message $m = b_1 b_2 b_3 \cdots b_k$, $x \in Z_N^*$, a set of representatives R for Z_N^* and an $N \in BI$, it is easy to compute the unique $x_0 \in Z_N^*$ such that $\sigma(x_0,N,R,m) = x$.

proof : The following algorithm computes x_0.

```
Algo B.
 input : x, N, R, m =(b_1, . . . , b_k)
 output : x_0 such that $\sigma(x_0,N,R,m) = x$
 begin
 if ( b_k =1 and $(\frac{x}{N}) \neq 1$) or ( b_k =0 and $(\frac{x}{N}) \neq -1$)
 then output   error (* x can not be an $(x_0,N,R)$-signature of m as $b_n \neq f_{b_n}(x^2)$ *)
 $x_k := x$
 for $j := k-1$ downto 0 do
      $x_j := x_{j+1}^2 \bmod N$.
      if ($b_j = 0$) and ($x_0 \geq \frac{N}{2}$)
      then $x_j = -x_j$
      else if $b_j = 1$ then
           if $x_j y < \frac{N}{2}$ set $x_j = x_j y \bmod N$
           else set $x_j = -x_j y \bmod N$
      end .
```

The running time of this algorithm is $k\,|N|^2$. QED

Lemma 10 : Given m $= b_1 b_2 \cdots b_k$, an $N \in BI$ (but not its prime factorization), $x_0 \in Z_N^*$, and $x \in Z_N^*$, it is easy to check whether x is the (x_0,N,R)-signature of m.

proof: if ($b_j = 1$ and $(\frac{x}{N}) = 1$) or ($b_j = 0$ and $(\frac{x}{N}) = -1$) then compute $u \in Z_N^*$ such that $\sigma(u,N,R,m) = x$ using algorithm B. Since such a u is unique, if $u = x_0$ then $x = \sigma(x_0,N,R,m)$. QED

3.5 The Signature Scheme

Each user A in a public key cryptosystem puts $P_A = (N_A, R_A, x_A)$ in the public file where N_A is a Blum-integer , $R_A = \{1, -1, y \in Q_3, -y\}$ is a set of representatives for N_A, and x_A is an element selected at random (with uniform probability) from $Z_{N_A}^*$. She keeps secret the prime factorization of N_A. That is $S_A = (p_A, q_A)$ where $N_A = p_A \cdot q_A$.

To sign the first message, m_1, A computes the (x_A, N_A, R_A)*-signature* of m_1, to sign the second message m_2, she computes the (x_A, N_A, R_A)*-signature* of $m_1 \cdot m_2$, where $\cdot$ denotes concatenation. and so on.

Comment 1 : The obvious disadvantage of this scheme is that the user who receives the signature of the ith message, also receives the signature of all the previous messages.

Comment 2: One way to avoid the problem discussed in comment 1 is to modify the scheme as follows. In addition to publicizing N_A and R_A , each user A publicizes a set of random quadratic residues mod N, $\{x_{1,A}, x_{2,A}, \ldots, x_{k,A}\}$. Suppose user A has already signed and sent messages m_1, m_2,..., m_t to user j. Let $P = m_1 m_2 \cdots m_t$. To sign the next message m for user j, user A computes $x := \sigma(x_{j,A}, N_A, R_A, P \cdot m)$. To verify that x is a $(x_{j,A}, N_A, R_A)$*-signature* of m, user j computes Y such that $\sigma(Y, N_A, R_A, P \cdot m) = x$, and checks that $Y = x_{j,A}$.

The drawback of this scheme is that k, the number of $x_{j,A}$'s in the public file, is as large as the number of users you are signing messages to.

Comment 3: Ron Rivest suggested the following modification, which is free of both of the above problems. User A publicizes $P_A = (N_A, R_A, x_A)$ and keeps secret the factorization of N_A, as described above. Let $c \geq 2$. First, user A picks at random from Z_N^* a sequence of starting locations, $x_A^1, x_A^2, \cdots, x_A^{2c} \in Z_{N_A}^*$, and signs them by computing an (x_A, N_A, R_A)*-signature* of $x_A^1 \cdot x_A^2 \cdots x_A^{2c}$. She then uses locations x_A^1 through x_A^c to sign messages m_1 through m_c, by computing the (x_A^i, N_A, R_A)*-signature* of m_i where $1 \leq i \leq c$. Each one of locations x_A^{c+1} through x_A^{2c} is used to sign a new list of $2c$ locations, out of which, in turn, the first c are used to sign messages, and the last c are used to sign additional lists of locations. Repeating this procedure recursively, we build a B-tree like directory for signatures. A signature of m will now consist of a list of signatures of directories, starting at x_A(which is in the public file) and ending at the signature of m, thus we can sign m without revealing the signatures of all the messages previously signed. Provided that the lists of starting locations are always picked at random from $Z_{N_A}^*$, this modified scheme can be easily proved(proof omitted) as secure as our initial scheme, whose security we prove in Theorem 11. Using this scheme, signing a message m still consists of $|m|$ square root extractions mod N. For $c=2$, to sign 2^{50} messages, the signature of m will be about $50n$ bits long.

3.6 The difficulty of forging signatures

Let M be a message space. Theorem 11 will show that if an adversary is capable of forging signatures of messages in M, she is able to factor hard integers. In order to appreciate the generality of theorem 1, let us discuss our model of an adversary.

First of all, we allow adversaries to choose which message m they try to forge, depending on the particular contents of the public file. Secondly, in real life a notary public signs contracts that are slightly different from each other. After seeing the digital signature of thousands of documents, an adversary could conceivably come up with one more similar contract whose signature he is able to forge.

Thus in our model we allow adversaries to try to forge signatures, having knowledge of the public file plus all the messages previously signed.

Finally, security is discussed in a probabilistic framework.

Let $0 < \varepsilon \leq 1$, $P(\cdot)$ be a polynomial function, and $M \subseteq \{0,1\}^*$ a message space with a given probability distribution . Let T be a polynomial time in n algorithm, that accepts as input $N \in BI$ such that $|N| = n$, $x_0^N \in Z_N^*$, a set of representatives for Z_N^*, $R^N = \{y_i \mid y_i \in Q_i\}$, a set of messages $m_1, m_2, \ldots, m_k \in M$ such that $\sum_{j=1}^{k} m_j \leq P(n)$, and $\sigma(x_0^N, N, R^N, m_1 \cdot m_2 \cdots m_k)$.

Theorem 11: If on inputs $N \in BI$, R^N, x_0^N, and $m_1, \ldots, m_k$, and $\sigma(x_0^N, N, R^N, m_1 \cdots m_k)$, with probability ε (averaged over the inputs R^N, x_0^N and $m_1, \cdots, m_k$), T outputs $m \in M - \{m_1, \ldots, m_k\}$, $\alpha \in \{0,1\}^*$ such that $|\alpha| < P(n)$ and $\sigma(x_0, N, R, \alpha \cdot m)$ (a legal signature of m), then there exists a probabilistic polynomial in n and ε^{-1} time algorithm, which factors $N \in BI$.

Sketch of the proof: Let T be a polynomial time algorithm which outputs the triplet m, x, α where $m \in M$, $\alpha \in \{0,1\}^*$ such that $|\alpha| < P(n)$, and $x \in Z_N^*$, on inputs N, $x_0^N \in Z_N^*$, set of representatives R^N, $m_1, \ldots, m_k \in M$ and $\sigma(x_0^N, R^N, N, m_1 \cdots m_k)$.

When m and x are such that $x = \sigma(x_0^N, R^N, N, \alpha \cdot m)$, then we say that T computed a good-output. We can quickly test whether $x = \sigma(x_0^N, R^N, N, \alpha \cdot m)$ by claim 7.

Let N be a member of BI of length n, such that for ε of the $x_0 \in Z_N^*$, the set of representatives for N, R, and k messages in M of length at most $P(n)$, T outputs a good- output. Then, the following algorithm, which uses T as an oracle, will factor N in time polynomial in $|N|$ and ε^{-1}

Pick an element b at random in Z_N^* such that $(\frac{b}{N}) = -1$. This can be done in expected time 2. b and -b are used as candidates for elements in Q_3 and Q_4 respectively. Since we do not know the factors of N, there is no known random polynomial time algorithm for picking elements of Q_3 or Q_4. But, we know that the pr $(b \in Q_3, -b \in Q_4) = \text{pr}(b \in Q_4, -b \in Q_3) = \frac{1}{2}$. So, either $\{1, -1, b, -b\}$ or $\{1, -1, -b, b\}$ is a set of representatives for N. Without loss of generality, let us assume that $\{1, -1, b, -b\}$ is a set of representatives for N.

Let us define an experiment :
Pick a set of k messages $m_1, \ldots, m_k$ in M at random such that $\sum_{j=1}^{k} m_j \leq P(n)$. Denote the binary representation of $m_1 \cdot m_2 \cdots m_k$ by $b_1 \cdots b_{P(n)}$. If $b_{P(n)} = 1$ then select z at random among all the elements in Z_N^* whose Jacobi symbol equals 1, otherwise select z at random among all the elements of Z_N^* whose Jacobi symbol equals -1. Choose $w \in Z_N^*$ at random and set $R = \{1, -1, bw^2, -bw^2\}$. Compute x_0 in Z sub N sup *' such that $\sigma(x_0, R, N, m_1 \cdots m_k) = z$ using algorithm B. There is a unique such x_0. As $m_1, \ldots, m_k$, z and R **were picked at random,** x_0 is also a random quadratic residue in Z_N^*. Input R, x_0, $m_1, \ldots, m_k$ and z to T. Three cases may occur.

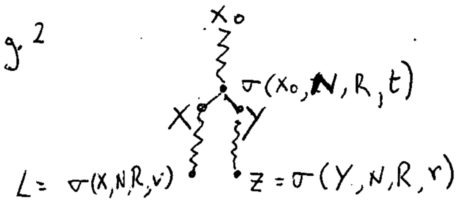
case 1: T outputs a bad-output.

case 2: T outputs (L, l, α) such that $l \in M$, $\sigma(x_0, R, N, \alpha \cdot l) = L$ and $m_1 \cdots m_k$ is a prefix of $\alpha \cdot l$. For this to happen T must compute a square root of z. But, since z was picked at random, T is essentially a random polynomial time procedure for extracting square roots mod N. By Rabin's theorem we can use T to factor N.

case 3: A outputs (L, l, α) such that $l \in M$, $\sigma(x_0, R, N, \alpha \cdot l) = L$ and some prefix of $m_1 \cdots m_k$ (includ-

ing the empty string) is a prefix of $\alpha \cdot l$.
Let $m_1 \cdots m_k = t \cdot r' \cdot r$, and $\alpha \cdot l = t \cdot v' \cdot v$, where $r', v' \in \{0,1\}$ and $r' \neq v'$. Then, compute X such that $\sigma(X,R,N,v)=L$ and Y such that $\sigma(Y,R,N,r)=z$. As $r' \neq v'$, $(\frac{X}{N}) \neq (\frac{Y}{N})$ and $X^2 \equiv Y^2 \bmod N$. See figure 2.

Fig 2



Thus, by facts 3 and 4, we can factor N.

By the law of large numbers we need to repeat this experiment only a polynomial in ε^{-1} number of times before we get a good output. If case 3 occurs, we factor immediately. If case 2 occurs we repeat the experiment using T as a square root extracting algorithm with success rate ε and factor N by Rabin's theorem. QED

4. A Strong Signature Scheme Based On The RSA Function

4.1 Background on RSA

Let $N = q_1 q_2$ be a product of two large primes, and $s > 1$ be an integer such that $gcd(s, \varphi(N)) = 1$; $\varphi(N)$ is the *Euler totient function*, and is equal to $(q_1 - 1)\cdot(q_2 - 1)$. The RSA encryption method [8] is based on the fact that it is easy to compute the s-th root $x^{\frac{1}{s}} \bmod N$ if one knows the factors q_1 and q_2, but appears intractable if the factors are unknown. For the RSA encryption to be effective, two types of assumptions are needed. Firstly, one needs to be able to generate efficiently the public-private key pairs. Secondly, the computing of the s-th root must be hard. We will now state a version of such assumptions.

The following method of generating a composite number N was suggested in [8]. Generate a random n-bit prime a_1; find the first prime in the arithmetic progression $1+2a_1, l+4a_1, 1+6a_1, \ldots,$ and call it $q_1 = 1+2d_1 a_1$. Repeat the above process to obtain a second random prime $q_2 = 1+2d_2 a_2$. Let $N = q_1 q_2$. Let us call the above the *standard procedure* for generating a composite N with parameter n. The next assumption will guarantee that the procedure terminates in polynomial time. For definiteness, we assume that one seldom needs to search more than n^3 numbers in finding q_1 and q_2.

Assumption 1: With probability $1 - o(1), d_1, d_2 \leq n^3$ as n goes to infinity.

Let $Z_N^* = \{y \mid 1 \leq y < N, gcd(y,N) = 1\}$. Let $C_{\varepsilon,n}$ be the size of the smallest boolean circuit C that takes as inputs three integers: a n-bit hard integer N, a prime $1 < s < N$, and $x \in Z_N^*$, and outputs an n-bit integer y such that $y = x^{\frac{1}{s}} \bmod N$ for a fraction ε of the inputs (N,s,x).

Assumption 2: For any fixed $\varepsilon > 0$ and any fixed polynomial $q(n), C_{\varepsilon,n} > q(n)$ as n goes to infinity.

4.2 Properties of the RSA function

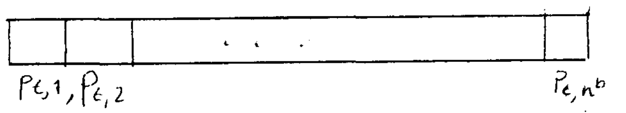
Fact 12: Let $N = p_1 p_2$, s such that $(s, \varphi(N)) = 1$ and $0 < \varepsilon \leq 1$. If there exists a polynomial time algorithm to extract s-roots mod N for a fraction ε of the x's in Z_N^*, then there exists a probabilistic polynomial in $|N|$ and ε^{-1} time algorithm that extracts s-th roots for all x's in Z_N^*.
Proof: Suppose there exists an algorithm A that accepts $x \in Z_N^*$ such that $x = y^s \bmod N$ as input, and outputs y for ε of the x's in Z_N^*. Pick $x \in Z_N^*$. Consider the following probabilistic algorithm for computing y such that $x = y^s \bmod N$: Pick r_i in Z_N^* at random. Compute A $[x r_i^s]$. By the law of large number, we need to repeat this only a polynomial in ε^{-1} number of times before A answers correctly. Suppose A answers correctly. Then, $y := A[x r_i^s]\ r_i^{-1} = y r_i r_i^{-1} \bmod N$. Note that computing the inverse of r_i mod N is easy.

Fact 13 (Shamir[12]): Let $0 < \varepsilon \leq 1$, n a positive integer, and $P(\cdot)$ be a fixed polynomial. Let $C_{\varepsilon,n,P}$ be the minimum size of a circuit C such that on inputs N, an n-bit hard integer, $x \in Z_N^*$, $s_1, \ldots, s_{P(n)}$ such that $(s_i, \varphi(N)) = 1$ and $(s_i, s_j) = 1$ for all $1 \leq i,j \leq P(n)$ and $\{y_i \mid y_i^{s_i} = x \bmod N, 2 \leq i \leq P(n)\}$, outputs y_1 such that $x = y_1^{s_1} \bmod N$ for ε of the x's in Z_N^*, then there exists another circuit C'_n of size at most $|C_{\varepsilon,n,P}|$ + a polynomial in ε^{-1} that on inputs N of size n and $x \in Z_N^*$, computes y such that $x = y^s \bmod N$.

4.3 A New Signature Scheme Based on RSA

We will define a signature scheme R. Let n be the parameter of the signature scheme. Each user A will have public information $P_A = (N, x, t, b)$, where $N = q_1 q_2$ is a random composite number generated by employing the standard procedure with parameter n, t is an integer between N and $\frac{N}{2}$, x is a random element uniformly chosen from Z_N^* and b is a positive constant. The private information is $S_A = (q_1, q_2)$.

We now describe how A will sign a sequence of messages $m_1, m_2, \ldots, m_h$. Let us consider the first n^b primes greater than t. Let $p_{t,i}$ denote the i-th smallest prime greater than t. By assumption 1, for large enough n, $p_{t,n^b} < t + n^3 n^b$.
Let us associate with this n^b sequence of primes, a paper tape containing n^b cells. The ith cell will corresponds to $p_{t,i}$.

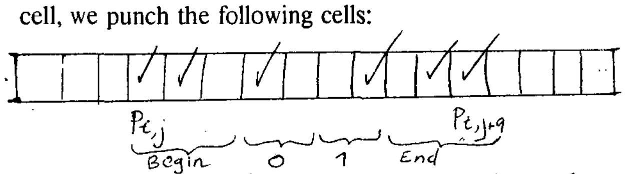


As it is easy to compute $p_{t,i}$ (by generating all the primes between t and $t + n^3 n^b$ and using a fast probabilistic primality testing algorithm[ ]), it is easy to reach cell i on the paper tape. To punch a hole in the ith cell of the tape, user A computes $x^{\frac{1}{p_{t,i}}} \bmod N$.

Now, to encode a k-bit string α on the tape, user A makes use of a block of $2k + 6$ unpunched consecutive cells on the tape. He must also leave 3 unpunched cells between any two signature blocks. He encodes α as follows:

Set $\alpha :=$ **begin** α **end** .
Encode **begin** by punching the first and second cells of the block and leaving the 3rd cell unpunched.
Encode **end** by punching the 2k+4th and 2k+5th cells of the block and leaving the 2k+3rd cell unpunched.
The $2i+2$ and the $2i+3$ cells of the block will be used to encode the ith bit: If the ith bit in α is 0, the $2i+2$ cell is punched and the $2i+3$ cell is left unpunched. Otherwise, if the ith bit in α is 1, the $2i+2$ cell is left unpunched and the $2i+3$ cell is punched.

For example, to encode α ='01" starting at the jth cell, we punch the following cells:



Thus, to sign a binary message m, user A encodes m in a suitably large, $2|m| + 6$ long, unpunched block of his tape. This segment of the tape is his signature of m. Note again that he must leave at least 3 unpunched cells in between every two signature blocks.

Theorem 12: R is an SSS

Sketch of the proof: To forge the signature of a message which was not previously signed, an adversary must punch at least one previously unpunched cell. But from fact 9 and 10 and assumption 2, it follows that even after seeing a polynomial number of previously punched cells, this is impossible for an adversary. Thus R is an SSS.

Remarks. We have omitted mathematical details in several places. A more comprehensive version, will appear in a forthcoming paper.

Acknowledgements

We are grateful to Danny Dolev and Avi Wiendgerson for a valuable discussion about signatures. We thank Joachim von zur Gathen for a remark about the distance between consecutive primes , Eric Bach for pointing us to the literature on consecutive primes, and Faith Fich for reading and commenting on a previous version of this paper. Finally, many thanks are due to Ron Rivest for finding the way to make the signature scheme based on factoring practical .

References

[1] M. Blum, "Coin flipping by telephone", *Proc. of IEEE, Spring Comp Con.* 1982, 133-137.

[2] L. Blum, M. Blum, M. Shub, "A Simple Secure Pseudo Random Number Generator", Crypto 1982.

[3] R. DeMillo, N. Lynch, and M. Merritt, "Cryptographic protocols", *Proc. 14th Ann. ACM Symp. on Th. of Comp.*, San Francisco, California, May 1982, 383-400.

[4] W. Diffie and M.E. Hellman, "New directions in cryptography", *IEEE Trans. on Inform. Th.* 22 (1976), 644-654.

[5] S. Goldwasser and S. Micali, "Probabilistic encryption and how to play mental poker keeping secret all partial information", *Proc. 14th Ann. ACM Symp. on Theory of Computing*, May 1982, San Francisco, California, 365-377.

[6] S. Goldwasser, S. Micali, and P. Tong, "How to establish a private code on a public network", *Proc. 23rd Ann. IEEE Symp. on Found. of Comp. Sci.*, Oct. 1982, Chicago, Illinois.

[7] M. Rabin, "Digitalized signatures and public-key functions as intractable as factorization", in MIT/LCS/TR-212, MIT Technical Memo, 1979.

[8] M. Rabin, "Digitalized signatures", in *Foundations*

of Secure Computations, edited by R. DeMillo, D. Dobkin, A. Jones, and R. Lipton, Academic Press, 1978, 155-168.

[9] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems", *Comm. ACM* 21 (1978), 120-126.

[10] A. Shamir, "On the generation of cryptographically strong pseudo-random sequences", *ICALP* 1981.

[11[A. Yao, "Theory and Applications of Trapdoor Functions", *Proc. 23rd Ann. IEEE Symp. on Found. of Comp. Sci.*, Oct. 1982, Chicago, Illinois.