MIT/LCS/TR-212

Digitalized Signatures and Public Key Functions as Intractable as Factorization

Michael O. Rabin

This blank page was inserted to preserve pagination.

MIT/LCS/TR-212

DIGITALIZED SIGNATURES AND PUBLIC-KEY FUNCTIONS AS INTRACTABLE AS FACTORIZATION

Michael O. Rabin

January 1979

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LABORATORY FOR COMPUTER SCIENCE

CAMBRIDGE MASSACHUSETTS 02139

This empty page was substituted for a blank page in the original document.

DIGITALIZED SIGNATURES AND PUBLIC-KEY FUNCTIONS AS INTRACTABLE AS FACTORIZATION

by

Michael O. Rabin

Visiting Professor of Applied Mathematics, MIT
Professor of Mathematics, Hebrew University
Jerusalem, Israel

ABSTRACT. We introduce a new class of public-key functions involving a number $n = p \cdot q$ having two large prime factors. As usual, the key n is public, while p and q are the private key used by the issuer for production of signatures and function inversion. These functions can be used for all the applications involving public-key functions proposed by Diffie and Hellman [2], including digitalized signatures. We prove that for any given n, if we can invert the function $y = E_n(x)$ for even a small percentage of the values y then we can factor n. Thus as long as factorization of large numbers remains practically intractable, for appropriatly chosen keys not even a small percentage of signatures are forgerable. Breaking the RSA function [6] is at most as hard as factorization, but is not known to be equivalent to factorization even in the weak sense that ability to invert all function values entails

ability to factor the key. Computation time for these functions, i.e. signature verification, is several hundred times faster than for the RSA scheme in [6]. Inversion time, using the private key, is comparable. The almost-everywhere intractability of signature-forgery for our functions (on the assumption that factoring is intractable) is of great practical significance and seems to be the first proved result of this kind.

Key words. Public-key functions, Digitalized signatures, Factorization, Intractable problems.

INTRODUCTION

In their fundamental paper [2] Diffie and Hellman have shown how public key trap door functions can be employed for the solution of various problems arising in electronic mail, including the production of digitalized signatures. An example of a public-key function usable for digitalized signatures was given in the elegant paper [6] by Rivest, Adelman, and Shamir, who introduced a trap-door one-way function employing a number n factorable into a product $n = p \cdot q$ of two large primes. The decoding algorithm given in [6] for this function requires knowledge of the factors p, q of n. It is, however, conceivable that another decoding algorithm exists that does not involve or imply factorization of n. Thus, breaking this one-way function is at most as difficult as factorization, but possibly easier.

We present a different public key function which can be used for digitalized signatures, and all the other applications, in the same way as the above-mentioned function. The function in [6] is 1-1. Our function is four to one, but this causes only slight modifications in the applications.

For this new function we can prove that the ability to forge signatures or decode messages is equivalent to the ability to factor large numbers. In fact, for any given n, a signature forgery or inversion algorithm effective in just a small percentage of all cases, say one case in a thousand, already leads to a factorization of n. By inversion we mean finding for a number y in the range of E one of the x such that $E(x) = y$.

In view of the present-day intractability of the factorization problem, this fact lends substantial support to the viability of our public-key function. As long as it is impossible in practice to factor large numbers, it will be impossible for a fixed key to forge signatures even for a small percentage of all messages.

The fact that we are able to prove, on the assumption that factoring is hard, that for our function, for a fixed key n whose factorization is not given, inversion must be hard for almost all messages is of great significance. For other trap door functions it may be the case that even though worst case complexity or even average complexity are high, in say one percent of cases inversion is

easy. From a commercial point of view this would pose an unacceptable risk. For example, an adversary can randomly search by computer for messages useful to him, such as payment instructions, on which he can forge signatures. To the best of our knowledge, we have in this article the first example of an almost everywhere difficult problem of this type.

In addition, computation time for this function is several hundred times faster, and inversion when p,q are known, is about eight times faster than the corresponding algorithms in [6]. If we invert the RSA function by Chinese Remaindering, as we do here, then inversion time for the two functions are comparable.

Theorems 1 and 2 concerning the equivalence of square-root extraction with factorization, are perhaps also of independent number-theoretic interest.

1. THE PUBLIC-KEY FUNCTION

Let $n = p \cdot q$ be the product of two large primes p,q, and let $0 \leq b < n$.

DEFINITION 1: The function $E_{n,b}(x)$ is defined for $0 \leq x < n$ by $E_{n,b}(x) \equiv x(x+b) \mod n$, $0 \leq E_{n,b}(x) < n$.

Computation of $E(x)$, for fixed n,b, requires one addition, one multiplication, and one division of

$x(x+b)$ by n to find the residue $E_{n,b}(x)$. Note that only the public key n,b, but not the factorization $n = p \cdot q$, is required for encoding.

2. INVERSION ALGORITHMS

Given $c \equiv x(x+b) \bmod n$, we want to find the four values $0 \leq x_i < n$, $1 \leq i \leq 4$ such that $E(x_i) = c$. We assume of course that the private key, i.e. the factors of n, are known.

Throughout this paper $res(A,B)$ will denote the residue of A when divided by B, and (A,B) will denote the greatest common divisor (g.c.d.) of A and B.

The decoder, who is the issuer of the public key n,b, knows the factorization $n = p \cdot q$. Clearly, it sufficies to solve the equation $x(x+b) \equiv c$ separately $\bmod p$ and $\bmod q$ and then find a solution $\bmod n$.

Let a be an integer so that $a \equiv 1 \bmod p$, $a \equiv 0 \bmod q$, and b satisfy $b \equiv 1 \bmod q$, $b \equiv \bmod p$. If r and s satisfy the congruence $\bmod p$ and $\bmod q$ respectively, then $z = ar + bs$ solves the congruence $\bmod n$, and $x = res(z,n)$ is the sought-after solution.

In what follows let p be a fixed prime. We shall understand all integers $\underline{a}$ to be residues mod p, i.e., $0 \leq a < p$. For d a quadratic residue (q.r.) mod p, $\sqrt{d}$ will denote any one of the two integers such that $(\sqrt{d})^2 \equiv \bmod p$, and $-\sqrt{d}$ will denote $p - \sqrt{d}$.

To solve

$$(1) \qquad f(x) = x^2 + bx - c \equiv \bmod p$$

let $d = b/2 \bmod p$ then $(x+d)^2 \equiv c + d^2 \bmod p$, $x = -d \pm \sqrt{c+d^2}$. We can solve the equation (1) as soon as we can extract square roots mod p, i.e., solve $y^2 - m \equiv 0 \bmod p$.

Assume first that $p = 4k - 1$ so that $4 \mid (p+1)$. Since m is a q.r., $m^{\frac{p-1}{2}} \equiv 1 \bmod p$. We claim that

$$(2) \qquad \ell = \sqrt{m} \equiv m^{\frac{p+1}{4}} \bmod p$$

is one of the two square roots of m. Namely,

$$\ell^2 \equiv m^{\frac{p+1}{2}} \equiv m \cdot m^{\frac{p-1}{2}} \equiv m \bmod p .$$

Thus one implementation of the function would use p and q such that $p \equiv q \equiv 3 \bmod 4$, and the decoding algorithm (2).

For $p = 4k + 1$ we directly solve the equation (1) by a probabilistic algorithm. This is a special case of Berlekamp's root-finding in GF(p) algorithm given in [1].

The short proof given here is taken from [5], where generalizations to $GF(p^n)$ appear. If the roots of (1) are $\alpha, \beta \in GF(p)$ then $x^2 + b\,x - c = (x - \alpha)(x - \beta)$ The roots in $GF(p)$ of the polynomial equation $x^{\frac{p-1}{2}} - 1 = 0$ are exactly the quadratic residues $\alpha \in GF(p)$. Consequently, if α is a quadratic residue while β is not, then $(x^{\frac{p-1}{2}} - 1, f(x)) = x - \alpha$, so that α and subsequently $\beta = -(b+\alpha) \bmod p$ are readily found.

Assume that α and β are of the same type, i.e., both quadratic residues (q.r.) or both quadratic non-residues mod p, and that $\alpha \neq \beta$. Let $0 \leq \delta < p$ then $\alpha + \delta$ and $\beta + \delta$ are of the same type if and only if $(\alpha+\delta)/(\beta+\delta)$ is a q.r. mod p. As δ takes all values $0 \leq \delta < p$ except $\delta = -\beta$, the quotient $(\alpha+\delta)/(\beta+\delta)$ takes all values $0 \leq \gamma < p$ except $\gamma = 1$. Thus for exactly $\frac{p-1}{2}$ choices δ, $\alpha+\delta$ and $\beta+\delta$ will not be of the same type.

Since $f(x-\delta) = (x-\alpha-\delta)(x-\alpha-\beta)$, we have that for a *random* choice of $0 \leq \delta < p$, with probability 1/2

$$(3) \quad (x^{\frac{p-1}{2}} - 1, f(x-\delta)) = x - \alpha - \delta \quad \text{or} \quad x - \beta - \delta.$$

Thus on the average two values of δ have to be tried for finding the roots of (1).

The computation of the g.c.d. (3) requires $O(\log_2 p)$ operations in GF(p), i.e., additions and multiplications

mod p. Namely, by essentially repeated squarings starting with x, compute $x + h = \mathrm{res}(x^{\frac{p-1}{2}}, f(x-\delta))$. Whenever a quadratic polynomial is encountered, divide by $f(x-\delta)$ to produce a linear polynomial. Note that x is a formal variable so that all computations involve just pairs of residues mod p. Now by (3), $x + h - 1$ is $x - \alpha - \delta$ or $x - \beta - \delta$, so that $-\delta - h + 1$ is a root of (1).

3. USE IN SIGNATURES

To employ E for signatures the signer P produces two large primes p,q by use of one of the prime-testing algorithms [3,7]. He forms $n = p \cdot q$, chooses a number $0 \leq b < n$ and publicizes the pair (n,b) (but not the factors p,q).

By convention, when wishing to sign a given message, M,P adds as suffix a word U of an agreed upon length k. The choice of U is randomized each time a message is to be signed. The signer now compresses $M_1 = MU$ by a hashing function to a word $C(M_1) = c$, so that as a binary number $c \leq n$; see [4]. The computation of C( ) is publicly known, so that $c = C(M_1)$ is checkable by everybody.

P now checks whether for this c the congruence

$$x(x+b) \equiv c \bmod n \tag{4}$$

is solvable.

By the analysis of Section 2, this congruence is solvable if and only if $m = c + d^2$ is a q.r. mod p and mod q. Thus testing the solvability of (4) amounts to computing the Jacobi Symbols $(\frac{m}{p})$ and $(\frac{m}{q})$ which is essentially a g.c.d. type computation.

If congruence (4) is not solvable then P picks another random U_1 and tries $c_1 = C(MU_1)$. The expected number of tries is 4. When for some U the congruence (4) is solvable for $c = C(MU)$, P finds a solution x.

DEFINITION 2: For a given public key n,b used by P and an agreed upon compressing function C( ) and integer k, P's signature on a message M is a pair U,x where $\ell(U) = k$ and $x(x + b) \equiv C(MU) \bmod n$.

Anybody can check P's signature by computing $c = C(MU)$ and testing whether $x(x+b) \equiv c \bmod n$.

The randomization of the suffix U of M also adds protection against possible attacks on the function E. Without the suffix, an adversary may attempt to feed to P messages M for his signature, hoping to learn the factorization of n from the solution of $x(x+b) \equiv C(M) \bmod n$, which will be produced by P as his signature. Actually, this does not seem a serious threat because of the hashing effected by C(M).

However, the randomized suffix of length k leads to essentially 2^k possible random values for $c = C(MU)$. Thus for, say, $k = 60$, the adversary has no effective control over the congruence (4) that P will solve.

4. INVERSION IS EQUIVALENT TO FACTORIZATION

We now want to show that if an adversary can invert $E_{n,b}(x)$ by any algorithm then he can factor n. By inverting we mean finding for y <u>one</u> of the four x such that $E_{n,b}(x) = y$. Finding one such x is sufficient for the would be signature forger, so that we want to show that this is hard. Thus the problem of, say, forging P's signatures is exactly as intractable as the factorization of a number n which is a product of large primes. As mentioned in the Introduction, the scheme in [6] is *at most* as safe as factorization but conceivably easier to crack.

In the following theorem we count an addition of numbers $a,b, \leq n$ as one operation.

It is readily seen that if we can solve (4) for fixed n,b and arbitrary c then we can extract square roots, i.e., solve $y^2 \equiv m \bmod n$ whenever a solution exists. Namely, letting $b \equiv 2d \bmod n$(n is odd) and $m = c + d^2 \bmod n$, (4) turns into

$$x^2 + 2dx + d^2 = (x+d)^2 \equiv m \bmod n.$$

Thus our result follows from

THEOREM 1: Let AL be an algorithm for finding one of the solutions of

$$y^2 \equiv m \bmod n \tag{5}$$

whenever a solution exists, and requiring $F(n)$ steps. There exists an algorithm for factoring n requiring $2F(n) + 2\log_2 n$ steps.

Proof. Assume that $n = p \cdot q$ is a product of two primes, the case relevant for $E_{n,b}$. The proof easily extends to the general case.

For any $0 < k < n$, $(k,n) = 1$, there are exactly four solutions for the congruence

$$y^2 \equiv k^2 \bmod n.$$

Namely, let $\mathrm{res}(k,p) = r$, $\mathrm{res}(k,q) = s$ then the solutions y of this congruence satisfy $\mathrm{res}(y,p) \equiv \pm r \bmod p$, $\mathrm{res}(y,q) = \pm s \bmod q$ and each of the four sign combinations gives rise to a different solution. Defining for $0 \leq y_1, y_2 < n$, $y_1 \sim y_2$ to mean $y_1^2 \equiv y_2^2 \bmod n$, we see that this equivalence relation decomposes the set $0 < y < n$, $(y,n) = 1$ into classes each containing four elements.

Denote by $\sqrt{m}$ the solution of (5) by AL for any m, $(m,n) = 1$. If AL produces more than one solution then

the factorization algorithm that follows is even further facilitated.

Choose at random a number $0 < k < n$. If $(k,n) \neq 1$ then we directly get a factor of n. In practice, this possibility can be neglected. Compute $k^2 \equiv m \bmod n$.

Compute $k_1 = \sqrt{m}$ by AL. Now, k is in the equivalence class, by the relation $\sim$, of k_1. In a random choice of $0 < k < n$, all four possible choices of numbers within any class are equally likely. Hence with probability 1/2

$$k \equiv k_1 \bmod p,\ k \equiv -k_1 \bmod q$$

or

$$k \equiv -k_1 \bmod p,\ k \equiv k_1 \bmod q$$

Therefore with probability 1/2

$$(k-k_1,n) = p \text{ or } q. \tag{6}$$

The computation of $\sqrt{m}$ requires $F(n)$ steps. The computation of the g.c.d. (6) requires at most $\log_2 n$ subtractions and divisions by 2, of numbers smaller than n. Hence the expected number of steps is $2F(n) + 2\log_2 n$.

If we count bit-operations then subtraction of numbers smaller than n requires at most $\log_2 n$ bit-operations and the bound is $2F(n) + 2(\log_2 n)^2$.

The previous theorem may be strengthened to cover the situation that for the given key $E_{n,b}$ can be decoded in just a small percentage of all cases.

THEOREM 2: If AL solves (5) in $F(n)$ steps for $1/e$ of the $0 < m < n$, $(m,n) = 1$, for which (5) has a solution, then there is an algorithm for factoring n requiring $2eF(n) + 2\log_2 n$ steps.

Proof. As in the proof of Theorem 1, choose a $0 < k < n$ at random and compute $k^2 \equiv m \bmod n$. Apply AL to find $\sqrt{m}$. If the computation runs more than $F(n)$ steps abort it and choose another k. Whenever a root $k_1 = \sqrt{m}$ is found, compute $(k-k_1,n)$. The analysis in the proof of Theorem 1 implies that with probability $1/2$ each such try produces a factorization of n.

The expected number of choices of k leading to a $\sqrt{m}$ is 2 and the expected number of successes of AL needed for a factorization, is 2. Thus the total expected number of steps is $2eF(n) + 2\log_2 n$. Note that we embark on the second phase of the factorization only after a success of AL in finding $\sqrt{m}$.

If for example $e = 1000$, and $F(n)$ were not prohibitively large, then an adversary could factor n in $2000\ F(n) + 2\log_2 n$ steps. Consequently, if no practical algorithm for factoring n is possible, then no practical decoding algorithm could work in even $1/1000$ of all cases.

5. GENERALIZATIONS

The above method of construction of a one-way function can be extended to employ polynomials or powers of x of small degrees other than 2.

Assume for example that $n = p \cdot q$, where p and q are primes of the form $3k + 1$. The one-way function will be $E(x) \equiv x^3 \bmod n$. The decoding is effected by solving $x^3 - m \equiv 0 \bmod p$ and $\bmod q$ by a probabilistic algorithm similar to the one used in Section 2. Again one can prove that any algorithm for extracting cubic roots leads, for n of the above form, to a factorization of n.

The probability that $x^3 \equiv w \bmod n$ is solvable for a random w is 1/9. Thus for utilization in signatures the quadratic scheme seems best.

REFERENCES

[1] Berlekamp, E.R., Factoring polynomials over large finite fields, Math. of Comp., vol. 24 (1970), pp. 713-735.

[2] Diffie, W., and Hellman, M., New directions in cryptography, IEEE Trans. of Inf. Theory, IT-22, (November 1976), pp. 644-654.

[3] Rabin, M., O., Probabilistic algorithms, Algorithms and Complexity, Recent Results and New Directions, J.F. Traub, editor, Academic Press, New York, 1976, pp. 21-40.

[4] Rabin, M.O., Digitalized signatures, Foundations of Secure Computations, R. Lipton and R. DeMillo editors, Academic Press New York, 1978.

[5] Rabin, M.O., Probabilistic algorithms in finite fields, submitted for publication.

[6] Rivest, R.L., Shamir A., and Adleman L., A method for obtaining digital signatures and public-key cryptosystems, Comm. ACM. vol. 21 (February 1978).

[7] Solovay, R., and Strassen, V., A fast monte-carlo test for primality, SIAM Jour. on Comp. Vol. 6 (1977), pp. 84-85.