AN EFFICIENT SIGNATURE SCHEME BASED ON QUADRATIC EQUATIONS

H. Ong
Mathematics Department
University Frankfurt
6000 Frankfurt am Main
Germany, FR

C.P. Schnorr[1]
Mathematics Department
University Frankfurt
6000 Frankfurt am Main
Germany, FR

A. Shamir[2]
Applied Mathematics Department
The Weizmann Institute of Science
Rehovot 76100
Israel

ABSTRACT

Electronic messages, documents and checks must be authenticated by digital signatures which are not forgeable even by their recipients. The RSA system can generate and verify such signatures, but each message requires hundreds of high precision modular multiplications which can be implemented efficiently only on special purpose hardware. In this paper we propose a new signature scheme which can be easily implemented in software on microprocessors: signature generation requires one modular multiplication and one modular division, signature verification requires three modular multiplications, and the key size is comparable to that of the RSA system. The new scheme is based on the quadratic equation $m = s_1^2 + ks_2^2 \pmod n$, where m is the message, s_1 and s_2 are the signature, and k and n are the publicly known key. While we cannot prove that the security of the scheme is equivalent to factoring, all the known methods for solving this quadratic equation for arbitrary k require the extraction of square roots modulo n or the solution of similar problems which are at least as hard as factoring. A novel property of the new scheme is that legitimate users can choose k in such a way that they can sign messages even without knowing the factorization of n, and thus everyone can use the same modulus if no one knows its factorization.

Remark: The scheme is an improved version of the signature scheme described in "Signatures Through Approximate Representations by Quadratic Forms" by Ong and Schnorr[5]. It is mathematically simpler, computationally faster and cryptographically comparable to the security of the original scheme.

1. INTRODUCTION

The communication revolution made possible by the home computer and the global telecommunication networks is likely to change the way we bank, shop, work, vote, access information, send mail, conduct business, and do almost everything else. To get this revolution into high gear, methods must be found to ensure the privacy, integrity and authenticity of the electronic messages. The most promising approach to the problem was proposed in Diffie and Hellman's seminal paper on public key cryptography in 1976. Since then, a number of cryptosystems have been developed and analysed, and the system which is currently believed to be the most secure is the RSA system which is based on the difficulty of factoring large numbers. Unfortunately, this system is very slow when implemented in software, and very expensive when im-

---

[1]Research supported by the Bundesminister für Forschung und Technologie under grant 08 30108

[2]This work was done while visiting the Computer Science Department of the University of Frankfurt

plemented in hardware with standard components. Until VLSI implementations become available, the practical uses of the RSA system will be severely limited.

The purpose of this paper is to describe a novel signature scheme which is somewhat related to the RSA system but which is hundreds of times faster. It can be easily implemented in software on microprocessors, and it is ideally suited to cheap, mobile items such as smart cards, identity tags, cellular telephones, data collection terminals, portable computers, and remotely controlled devices, where authentication and signature schemes are essential. If it can be shown to be secure (or at least if it withstands concentrated cryptanalytic attack for a reasonable period), it can be a truely practical and cost-effective solution for almost any unforgeable authentication problem.

2. THE SIGNATURE SCHEME

When Alice joins the communication network, she publishes a key consisting of two parts: a multiplier k and a modulus n. Both parts are large numbers (at least 1000 bits long), and n is a composite number whose factorization is unknown (except possibly to Alice). The messages m are numbers in the range $0 \le m < n$, and their signatures are pairs of numbers s_1, s_2 in the same range. The signature is considered valid if the following modular equation holds:

$$(1) \qquad m = s_1^2 + k s_2^2 \pmod{n} .$$

If Bob knows k and n, he can easily verify Alice's signature by performing three modular multiplications and one modular addition.

Unlike the RSA system, signatures are not uniquely associated with messages. Since the number of possible messages is n while the number of possible signature pairs is n^2, each message has about n different signatures. However, the probability that a randomly chosen pair s_1, s_2 will be a valid signature of a given m is negligible, and thus the multiplicity of signatures does not imply that they are easy to find.

How does Alice choose her public key? She first chooses the modulus n as a large composite number which is difficult to factor. By using a probabilistic primality testing algorithm on random integers with at least 500 bits, Alice can find after a few hundred tests two numbers p and q which are almost certainly primes. The product n of p and q is easy to compute, but even the fastest known factoring algorithm on the fastest available computer will take millions of years to factor it.

Efficient implementations of this process can compute n within a few hours on a typical microcomputer. Such an overnight initialization period is acceptable in most applications, but if the user cannot afford it, there is a faster alternative: If a trusted third party (the NBS?) computes n and then erases p and q, no one knows the factorization of n and thus everyone can use it as a standard modulus.

The secret that helps Alice solve the quadratic equation (1) for any m is a number u which is relatively prime to n and which satisfies:

$$(2) \qquad 1 + k u^2 = 0 \pmod{n} .$$

It is very easy to compute k from u, but very difficult to recover u from k since square roots cannot be extracted without knowing the factorization of the modulus n. Alice follows the easy direction during the key generation phase by picking u at random and computing k as:

$$(3) \qquad k = -1/u^2 \pmod{n} .$$

Once k is published, Bob (or anyone else) cannot compute u, and thus cannot follow the signature generation method that Alice is using.

Since u is relatively prime to n, any pair of numbers s_1, s_2 between 0 and n-1 can be expressed as

$$(4) \qquad s_1 = t + r \pmod{n} , \quad s_2 = t \cdot u \pmod{n}$$

for appropriately chosen values of t and r (use $t = u^{-1} \cdot s_2 \pmod{n}$, $r = s_1 - t \pmod{n}$) .

If s_1 and s_2 are a solution of equation (1), we can substitute the expressions in (4) into (1) and get:

(5) $\quad m = (t+r)^2 + k(tu)^2 \pmod n$

which can be simplified to

(6) $\quad m = t^2(1+ku^2) + t(2r) + r^2 \pmod n$.

By the definition of k, the coefficient of t^2 is zero, and thus (6) becomes a linear equation in t which can be easily solved:

(7) $\quad t = (\frac{m}{r} - r)/2 \pmod n$.

Substituting this value of t into (4), we get the fundamental equations

(8) $\quad s_1 = (\frac{m}{r} + r)/2 \pmod n$, $\quad s_2 = (\frac{m}{r} - r) \cdot u/2 \pmod n$

Alice can thus generate signatures of m by choosing a random r and evaluating (8), using one modular multiplication, one modular division, two modular additions/subtractions, and two trivial divisions by 2. If r is not relatively prime to n, m/r (mod n) may not be defined, but if all the factors of n are large Alice is unlikely to choose such an r.

The relationships between messages and signatures are summarized by the following lemma:

Lemma 1: Let $\mathbb{Z}_n^*$ be the set of numbers between 0 and n which are relatively prime to n, and let m be a fixed element in $\mathbb{Z}_n^*$. Then the set of signatures of m is in 1-1 correspondence with the set of values of (8) as r ranges over $\mathbb{Z}_n^*$.

Proof: For each $r \in \mathbb{Z}_n^*$, (8) is clearly a signature. For every signature, there are t and r satisfying (6), but if $r \notin \mathbb{Z}_n^*$ the right hand side of (6) is not in $\mathbb{Z}_n^*$, which contradicts the assumption on m. Since $s_1 - s_2/u = r \pmod n$, only one value of r in $\mathbb{Z}_n^*$ can correspond to each signature. Q.E.D.

Remarks: (i) By using a random r, Alice can choose an arbitrary signature of m with uniform probability distribution, and is not restricted to signatures of some special form.

(ii) If two different messages are signed with the same r, u can be computed from the signatures and thus Alice must choose a new random r for each message.

(iii) Messages which are not relatively prime to n should not be signed. However, only $m = 0$ should be explicitly excluded since any other message of this type is unlikely to occur.

The various components of the signature scheme can be summarized as follows:

Key Generation:

1. Pick a large composite number n.
2. Pick a random u which is relatively prime to n.
3. Compute $k = -1/u^2 \pmod n$.
4. Publish n,k, and keep u secret.

Signature Generation ($m \neq 0$):

1. Pick a random r which is relatively prime to n.
2. Compute $s_1 = (m/r+r)/2 \pmod n$
   $s_2 = (m/r-r) \cdot u/2 \pmod n$.

Signature Verification:

1. Compute $s_1^2 + k \cdot s_2^2 \pmod n$.
2. If the result is m, the signature is valid.

3. SECURITY CONSIDERATIONS

The security of the scheme relies on Bob's inability to gain information from Alice's signatures (proved in this section) and on the difficulty of solving the binary quadratic equation (1) (discussed in the next section).

Bob cannot possibly discover the factorization of n by analyzing large collections of (m, s_1, s_2) triplets produced by Alice, since even Alice may be unaware of it and thus cannot reveal it accidentally. From her point of view, all her actions consist of random number generation and the evaluation of simple formula, and thus they cannot lead to the factorization of n if factoring is difficult.

The next cryptanalytic attack is related to equations (8). Given h message-signature triplets (m^i, s_1^i, s_2^i) produced by random r^i values, Bob gets 2h equations in $h+1$ variables (u and the r^i). Could he manipulate these equations in some way and compute some of these unknowns? The following

theorem answers this question negatively even when the m^i are chosen by Bob or when the same message is repeatedly signed:

Theorem 2: Any algorithm for computing u from random signatures of messages of its choice can be transformed into a probabilistic factoring algorithm with similar complexity.

Proof: When n has at least two distinct odd factors, the number -1/k has at least four distinct square roots. By Lemma 1, all these roots create the same set of signatures (in a different order, but with the same probability distribution) when r varies and m is relatively prime to n. The algorithm that analyzes the signatures thus cannot distinguish between the roots and cannot identify the root u which is actually used in the signature generation process. If this root u has been chosen at random then the probability that the root u' reported by the algorithm satisfies $u' \neq \pm u \pmod n$ is at least one half, and in this case gcd(u-u',n) is an non-trivial factor of n.

In order to factor a given number n, Bob picks a signature scheme by choosing u at random and computing the appropriate k. Bob then signs the messages requested by the algorithm for the key (k,n), and compares the u' reported by the algorithm with the u he knows. By repeating this process sufficiently many times with the same n but different u values, n will be factored with arbitrarily high probability of success. Q.E.D.

Remarks: (i) If Bob could compute some r^i value, he could substitute it into the expression for s_2 in (8) and isolate u. The r^i values are thus as hard to compute as u.

(ii) The theorem can be easily extended to the case of an algorithm that succeeds for only some of the k, provided that the fraction of these k is non negligible.

(iii) In Rabin's signature scheme an opponent can factor n by analysing the signatures of specific messages. In our scheme the factorization of n and the secret parameter u cannot be revealed by chosen message attacks.

Since Bob cannot benefit from Alice's signatures and cannot use her method for solving equation (1), he must come up with an alternative way of solving this equation. The known methods for solving binary quadratic equations are summarized in the next section.

4. THE COMPLEXITY OF SOLVING $s_1^2 + ks_2^2 = m \pmod n$.

Equation (1) has a number of symmetries and closure properties, which seem to be useful both in proofs of security and in cryptanalysis. They are based on the following facts:

(i) The roles of k and m in equation (1) can be interchanged, since a signature s_1, s_2 of message m with key (k,n) yields a signature $s_1 s_2^{-1} \pmod n$, $s_2^{-1} \pmod n$ of message -k with key (-m,n).

(ii) Signatures s_1', s_2' of m' and s_1'', s_2'' of m'' with the same key (k,n) yield a signature s_1, s_2 of $m = m'm'' \pmod n$ by composition (see [10], [11] for definitions and background material):

$$(9)\quad s_1 = s_1's_1'' - ks_2's_2'' \pmod n,\quad s_2 = s_1's_2'' + s_2's_1'' \pmod n .$$

The validity of the signature of m follows from the identity

$$(10)\quad (s_1'^2 + ks_2'^2)(s_1''^2 + ks_2''^2) = (s_1's_1'' - ks_2's_2'')^2 + k(s_1's_2'' + s_2's_1'')^2 \pmod n$$

(iii) For a fixed signature s_1', s_2' of $m' \in \mathbb{Z}_n^*$, equation (9) implies that the signatures s_1'', s_2'' of m'' are in 1-1 correspondence with the signatures s_1, s_2 of $m = m'm'' \pmod n$, since the system of linear equations is non singular with determinant $s_1'^2 + ks_2'^2 = m' \pmod n$. This proves that for $m \in \mathbb{Z}_n^*$ the number of solutions of equation (1) is independent of m even when -k is a quadratic non residue mod n.

(iv) An important special case of properties (i) and (ii) is that a signature s_1, s_2 of m with (k,n) yields a signature $vs_1 \pmod n$, $vs_2 \pmod n$ of $v^2m \pmod n$ with key (k,n), and a signature s_1, $v^{-1}s_2 \pmod n$ of m with key $(v^2k \pmod n, n)$, for any $v \in \mathbb{Z}_n^*$.

A simple consequence of these properties is that the signature scheme has uniform complexity: Either

all or almost none of the pairs of messages m and multipliers k can be broken. More precisely, we prove:

Theorem 3: Any T(n)-time algorithm which for each n solves equation (1) for an ε-fraction of the messages m and multipliers k can be transformed into a probabilistic $O(T(n)/\varepsilon)$ time algorithm which with probability $\geq 1/2$ solves equation (1) for all the n, m and k.

Proof: By properties (i) (ii) and (iii), it is possible to change m into any other $\bar{m}$, and k into any other $\bar{k}$, with uniform probability distribution so that each signature $\bar{s}_1$, $\bar{s}_2$ of $\bar{m}$ with $(\bar{k},n)$ can be changed back into a signature s_1, s_2 of m with (k,n) in a fixed number of arithmetic operations. By repeating this randomization of m, k into $\bar{m}$, $\bar{k}$ at least $1/\varepsilon$ times, a valid signature is obtained with probability $\geq (1-(1-\varepsilon)^{1/\varepsilon}) \geq 1-e^{-1} \geq 1/2$.

Q.E.D.

Equation (1) is a special case of the general binary quadratic equation

$$(11)\quad a_1 s_1^2 + a_2 s_2^2 + a_3 s_1 s_2 + a_4 s_1 + a_5 s_2 + a_6 = 0 \pmod n,$$

but in fact the two equations have similar complexity:

Lemma 4: Any polynomial time algorithm for solving equations of type (1) can be transformed into a probabilistic polynomial time algorithm for solving equations of type (11).

Proof: There is a straightforward linear substitution to s_1, s_2 which transforms almost every equation (11) into an equation of type

$$\bar{a}_1 \bar{s}_1^2 + \bar{a}_2 \bar{s}_2^2 + \bar{a}_6 = 0 \pmod n .$$

Multiplication with $\bar{a}_1^{-1} \pmod n$ yields an equation of type (1). If the linear substitution is undefined we either find a factor of n, since some coefficients are not relatively prime to n, or equation (11) is transformed into an equation which is linear in at least one variable. In the latter case equation (11) is easy to solve. If n has been factored we can solve equation (11) by Rabin's probabilistic polynomial time square rooting algorithm.

Q.E.D.

The main reason we believe that signatures cannot be forged in our scheme is that all the known methods for solving binary quadratic equations modulo n are tantamount to factoring. The simplest method is to fix one of the variables s_i and solve the remaining single variable equation. However, the computation of either s_1 or s_2 requires the extraction of a square root (mod n) which is as difficult as factoring n.

Another approach might be to solve equation (1) over the integers, since any such solution (when reduced modulo n) gives a solution for the modular equation. However, an easy counting argument shows that for all m_o the fraction of $m \in [1,m_o]$ for which equation (1) is solvable is $O(1/\sqrt{k})$, so for large k this method is not likely to suceed. The next method for solving equation (1) exploits the theory of quadratic forms. The group $SL_2(\mathbb{Z})$ of 2×2 integer matrices T with $\det T = 1$ acts on the set of binary quadratic forms

$$ax_1^2 + bx_1x_2 + cx_2^2 = x^tAx , \quad A = \begin{bmatrix} a & b/2 \\ b/2 & c \end{bmatrix} ,$$

$$a,b,c \in \mathbb{Z} , \quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

via the transformation $A \to T^tAT$, where T^t is the transpose of T. We denote these forms by (a,b,c). Two forms (a,b,c) and (a',b',c') are called equivalent if there is some $T \in SL_2(\mathbb{Z})$ s.t. $T^tAT = A'$. Let $[(a,b,c)]$ be the equivalence class of (a,b,c). (a,b,c) represents $m \in \mathbb{Z}$ if there is some $s \in \mathbb{Z}^2$ s.t. $s^tAs = m$. Obviously $s^tAs = m$ iff $s'^tT^tATs' = m$ with $s' = T^{-1}s$. Equivalent forms represent precisely the same elements in $\mathbb{Z}$ and this suggests to consider quadratic equations and their solutions modulo $SL_2(\mathbb{Z})$. In the remainder of the paper we restrict ourselves to forms (a,b,c) which are primitive, i.e. gcd(a,b,c) = 1 and positive, i.e. $a > 0$ and have negative discriminant $\Delta := b^2 - 4ac = -4 \det A$. Note that the polynomial $x_1^2 + kx_2^2 = (1,0,k)$ with discriminant $-4k$ has these properties. Let $G(\Delta)$ be the set of equivalence classes $[(a,b,c)]$ with discriminant Δ.

Theorem 5 (Gauss 1801, Art. 234-244)

(1) $G(\Delta)$ forms an abelian group under composition of forms, the class group.

(2) $|G(\Delta)| = O(\sqrt{|\Delta|}\log|\Delta|)$ ($|G(\Delta)|$ is the order of the group $G(\Delta)$).

(3) For two equivalent forms a transformation matrix T can be found in polynomial time.

(4) The class $[(1,0,k)]$ is the unit element of $G(-4k)$.

(5) Composition of forms can be done in polynomial time.

It is important that the composition in $G(\Delta)$ is compatible with the multiplication of integers represented by the corresponding forms. That means if $s'^tA's' = m' \pmod n$, $s''^tA''s'' = m'' \pmod n$, and the composition of A' and A" yields A then we easily get from s', s" a vector $s \in \mathbb{Z}^2$ such that $s^tAs = m'm'' \pmod n$. In particular if $A' = A'' = \begin{bmatrix} 1 & 0 \\ 0 & k \end{bmatrix}$ then $A = \begin{bmatrix} 1 & 0 \\ 0 & k \end{bmatrix}$ and s are obtained from s', s" by the formulae (9).

The formulae of composition: A form (a,b,c) in the product class $[(a',b',c')][(a'',b'',c'')]$ is obtained from a',b',c', a",b",c" as follows (for explanation see Lenstra 1982)

$$
\begin{aligned}
&d := \gcd(a', a'', (b'+b'')/2)\\
&\text{Let } \alpha,\beta,\gamma \in \mathbb{Z} \text{ be such that}\\
&\alpha a' + \beta a'' + \gamma\frac{(b'+b'')}{2} = d\\
&a := a'a''/d^2\\
&b := b'' + \frac{2a''}{d}\left[\left(\beta\frac{b'-b''}{2} - \gamma c'\right) \bmod \frac{a'}{d}\right]\\
&c := (-\Delta + b^2)/(4a)
\end{aligned}
\tag{12}
$$

The equivalence class $[(a,b,c)]$ does not depend on the particular choice of α,β,γ. If m is represented by (a',b',c') and m" by (a",b",c") then m'm" can be represented by (a,b,c):

$$
(a's_1'^2 + b's_1's_2' + c's_2'^2)\,(a''s_1''^2 + b''s_1''s_2'' + cs_2''^2) = as_1^2 + bs_1s_2 + cs_2^2
$$

holds for

$$
\begin{aligned}
s_2 &:= \frac{1}{d}\left[(a's_1' + \frac{b'}{2}s_2')s_2'' + s_2'(a''s_1'' + \frac{b''}{2}s_2'')\right]\\
s_1 &:= \frac{1}{d}\left[(a's_1' + \frac{b'}{2}s_2')(a''s_1'' + \frac{b''}{2}s_2'') + \frac{\Delta}{4}s_2's_2''\right] - \frac{b}{2}s_2 .
\end{aligned}
\tag{13}
$$

s_1, s_2 are polynomials in s_1', s_2', s_1'', s_2'' with integer coefficients (note that $b' \equiv b'' \equiv b \pmod 2$). The integers a,b,c of the composed form are much larger than a',b',c',a",b",c". In order to keep numbers small it is important to reduce the composed form. The whole process of composition and reduction takes $O(\log|\Delta|)^2$ bit operations[11].

A key observation: $x_1^2 + kx_2^2 = m \bmod n$ can be solved efficiently if the order of $G(-4k)$ is odd and if all prime powers in the prime decomposition $|G(-4k)| = \prod_i p_i^{\nu_i}$ are small. For the purpose of this paper call $|G(-4k)|$ t-smooth if $\max_i p_i^{\nu_i} \le t$. Let QR_m be the set of squares modulo m.

Theorem 6: Let k, m be prime, $k \equiv -1 \bmod 4$, $-k \in QR_m$ and $|G(-4k)|$ t-smooth. Then by a probabilistic algorithm $x_1^2 + kx_2^2 = m \pmod n$ can be solved (with probability $\ge 1/2$) in $O(t + \log m)$ steps. (Multiplications of integers $\le n$ and compositions are counted as single steps).

Algorithm 7: (proof of theorem 6)

1. solve $b'^2 \equiv -4k \bmod 4m$ by Rabin's probabilistic square root algorithm (this requires that m is prime and $-k \in QR_m$ and takes $O(\log m)$ steps).
2. $c' := (b'^2 + 4k)/(4m)$
   (then (m,b',c') has discriminant -4k)
3. construct the sequence $p_1,\dots,p_l$ of odd primes $\le t$ $\quad e_i := \max\{\nu \mid p_i^\nu \le t\}$ $i = 1,\dots,l$
4. (It is known that $|G(-4k)|$ is odd for k prime, $k \equiv -1 \bmod 4$. Hence $|G(-4k)| \mid \prod_{i=1}^{l} p_i^{e_i}$ and $[(m,b',c')]^{\prod_{i=1}^{l} p_i^{e_i}} = [(1,0,k)]$)
   transform the solution $s_1' = 1$, $s_2' = 0$ of $ms_1'^2 + b's_1's_2' + c's_2'^2 = m$ into a solution s_1, s_2 of $s_1^2 + ks_2^2 = m^{\prod_{i=1}^{l} p_i^{e_i}} \pmod n$.
   (this takes $\le 2\cdot\log_2 \prod_{i=1}^{l} p_i^{e_i} \le 2l\log_2 t \le 4t$ compositions)
5. $\bar{s}_\nu := s_\nu^{-(\prod_{i=1}^{l} p_i^{e_i} - 1)/2} \pmod n$ for $\nu = 1,2$.
   (this takes $O(t)$ multiplications and one division in $\mathbb{Z}_n^*$, obviously $\bar{s}_1^{-2} + k\bar{s}_2^{-2} \equiv m \pmod n$).

Q.E.D.

Exploiting the ideas of the factoring algorithm of Schnorr and Lenstra[10], algorithm 7 can be extended to arbitrary multiplier-message pairs (k,m). By experience the class groups G(-4k) behave like "random abelian groups of order $O(\sqrt{k}\log k)$". The groups G(-4k) and G(-4kv) are unrelated provided that kv is squarefree. In our analysis we assume

that the fraction of "all $v \le k^{\varepsilon}$ for which $|G(-4kv)|$ is t-smooth" is at least the fraction of "all integers $\le k^{(1+\varepsilon)/2}$ which are t-smooth". The fraction of t-smooth integers is known from

Theorem 8: (Canfield, Erdös, Pomerance[1,7])
Let $\psi(n,v) := \#\{z \le n : z \text{ is free of primes} > v\}$ then for all $a, c > 0$: $\psi(n^c, L_n^a) n^{-c} = L_n^{-c/(2a)+o(1)}$. Here $L_n := \exp\sqrt{\ln n \ln\ln n}$ and $\lim_n o(1) = 0$.

Solving $x_1^2 + kx_2^2 = m \pmod n$ for small k.

(i) Since $|G(-4k)|$ is $k^{1/2+o(1)}$-smooth we can apply algorithm 7 with $t = k^{1/2+o(1)}$. This solves $x_1^2 + kx_2^2 = m \pmod n$ provided that m is prime, $-k \in QR_m$ and $[(m,b',c')] \in G(-4k)$ has odd order. These conditions on m can be satisfied by applying a few random transformations to m. Hence equation (1) can be solved in $k^{1/2+o(1)}$ steps.

(ii) We show how to transform a triple of solutions s_1', s_2' of $s_1'^2 + kvs_2'^2 = m \pmod n$, $\bar{s}_1, \bar{s}_2$ of $\bar{s}_1^2 + v\bar{s}_2^2 = m \pmod n$ and $\tilde{s}_1, \tilde{s}_2$ of $\tilde{s}_1^2 + \tilde{s}_2^2 = m \pmod n$ into a solution of $s_1^2 + ks_2^2 = m \pmod n$: Since $(\bar{s}_2/\bar{s}_1)^2 - m(1/\bar{s}_1)^2 = -1/v \pmod n$ and $(s_1'/s_2')^2 - m(1/s_2')^2 = -kv \pmod n$ composition of $\bar{s}_2/\bar{s}_1$, $1/\bar{s}_1$ and s_1'/s_2', $1/s_2'$ yields a solution s_1'', s_2'' of $s_1''^2 - ms_2''^2 = k \pmod n$. Since $(\tilde{s}_1/\tilde{s}_2)^2 - m(1/\tilde{s}_2)^2 = -1 \pmod n$ composition of s_1'', s_2'' and $\tilde{s}_1/\tilde{s}_2$, $1/\tilde{s}_2$ yields a solution s_1''', s_2''' of $s_2'''^2 - ms_2'''^2 = -k \pmod n$. Hence $s_1 := s_1'''/s_2'''$, $s_2 := 1/s_2'''$ solves $s_1^2 + ks_2^2 = m \pmod n$.

(iii) Our fastest method for solving equation (1) for small k has time bound $L_k^{1+o(1)}$, $L_k = \exp\sqrt{\ln k \ln\ln k}$. In this method we search for $v_o := \min\{v : |G(-4kv)| \text{ is } L_k^{1/2}\text{-smooth}\}$. The "random behaviour" of the groups G(-4kv) and theorem 9 imply $v_o \le L_k^{1/2+o(1)}$. $s_1'^2 + kv_o s_2'^2 = m \pmod n$ can be solved in $L_k^{1/2+o(1)}$ steps by applying algorithm 7 to a few randomly transformed versions of m. Trying all $v \le v_o$ takes $L_k^{1+o(1)}$ steps. $\bar{s}_1^2 + v_o\bar{s}_2^2 = mv_o \pmod n$ and $\tilde{s}_1^2 + \tilde{s}_2^2 = m \pmod n$ can be solved in time $L_k^{1/2+o(1)}$, see (i).

Applying (ii) we can easily compose a solution s_1, s_2 of $s_1^2 + ks_2^2 = m \pmod n$ from s_1', s_2', $\bar{s}_1, \bar{s}_2$ and $\tilde{s}_1, \tilde{s}_2$. The whole computation can be done in $L_k^{1/2+o(1)}$ steps.

Avoid small messages m: Since the roles of m and k in equation (1) can be interchanged, the case of small m is as easy as the case of small k. Small messages must be avoided (e.g. by expanding them to full size with an appropriate one-way function).

The fastest attack on the signature scheme: In our fastest method to solve $s_1^2 + ks_2^2 = m \pmod n$ for large k the cryptanalyst tries to transform k into some $\bar{k}$ which he can afford to factor completely, say $\bar{k} = \prod_i p_i^{\nu_i}$. He solves $s_{1,i}^2 + p_i s_{2,i}^2 = m \pmod n$ for all p_i in about $L_{p_{max}}$ steps, $p_{max} := \max_i p_i$. Using composition he combines these solutions and retracing the transformation of k he finds a solution s_1, s_2 of equation (1).
The fastest known algorithm which transforms k into some $\bar{k}$ such that $p_{max} \le n^{0.4}$ and $\bar{k}/p_{max}$ is $L_n^{9/\sqrt{10}}$-smooth, takes $L_n^{\sqrt{0.4}+o(1)}$ steps. (This algorithm uses Pollard's ρ-method[6] for detecting the small primes of $\bar{k}$ and the early abort strategy[7] for passing the useless $\bar{k}$ quickly.) For a 1000 bit modulus n the method takes about 2^{63} steps. Each step (multiplication or division of 1000 bit integers, or composition) itself consists of about 2^{20} bit operations. Moreover the attack is intricate and requires sophisticated programs. In spite of the optimistic assumptions in our analysis the method is clearly unfeasible if the modulus n has at least 1000 bits.

Summary: The methods for factoring n and for solving the equation $x_1^2 + kx_2^2 = m \pmod n$ are closely related. The fastest known algorithms have time bounds L_n for factoring n, see Schnorr, Lenstra[10] and $L_n^{\sqrt{0.4}}$ for solving equation (1), $L_n = \exp\sqrt{\ln n \ln\ln n}$. The time bounds are the same except for the constant in the exponent. The complexity of solving $x_1^2 + kx_2^2 = m \pmod n$ is comparable to the complexity of factoring integers of size $\sqrt{n}$. Therefore the modulus n for the new signature scheme should be about twice as long as for

the RSA-scheme.

5. APPLICATIONS, EXTENSIONS AND OPEN PROBLEMS

Commutative modular exponentiation functions (with the same modulus but different exponents) play an important role in many key management and communication protocols. The fact that in the new signature scheme everyone can use the same modulus (if its factorization is unknown) is likely to play a similar role in novel applications of signatures.

The idea of using a common n is particularly attractive when a large organization wants to create many signature keys for all its employees. If the value of n is computed under tight security by the headquarters of the organization, each employee can pick his own secret u which is not revealed to anyone else. Unfaithful employees cannot forge the signatures of other employees, and even security breaches at the headquarters do not endanger the system if they happen after p and q are erased.

A similar idea is applicable to large timesharing computer systems. During logon, the system generates a random message m and asks the user to sign it in order to prove his identity. This is much more secure than password authentication since an opponent cannot reuse an old signature and cannot forge a new signature even if he gains access to the file of authentication keys. If the value of n is computed once and made public by the system administrator, the computational burden on the users during key generation becomes minimal.

It is easy to extend the signature generation technique to equations with more than two variables or degrees higher than 2, see appendix A. It is also possible to use systems of equations where both the message and the signature are vectors of values, and the signature is considered valid only when all the equations are satisfied. However, it is not clear whether such schemes have any performance or security advantage over the simpler binary quadratic case considered in this paper.

It is not known whether a similar scheme can be used as a public key cryptosystem. The natural approach is to change the role of the variables - to make s_i the cleartext and m the ciphertext. However, without further restrictions on the values of the s_i, the cleartexts are not uniquely determined by the ciphertexts, and thus it is impossible to decrypt.

The main open problem concerning the new scheme is its security. To make the signature scheme as secure as possible, users should choose a large modulus n (to make its factoring difficult), change the value of k frequently (to make its analysis useless), perturb messages before they are signed (to prevent attacks based on multiplicativity), and expand short messages to full size (since short messages are easy to sign). However, the list of cryptanalytic attacks mentioned in Sections 3 and 4 is almost certainly incomplete, and the reader is encouraged to find new attacks and to analyze their feasibility.

APPENDIX A: A Simple Technique For Generating Trapdoor Polynomials

A general method for creating trapdoor functions with arbitrary number of variables is to start from an easily invertible function and then scramble its description to make it apparently difficult. To use this method, choose a polynomial $P'(x_1,\ldots,x_v)$ which is linear in x_1 with arbitrary total degree $d \geq 2$. All the solutions of the equation $P'(x_1,\ldots,x_v) \equiv 0 \pmod n$ can be obtained by picking arbitrary values for $x_2,\ldots,x_v$ and then solving the linear equation in the remaining variable x_1. To scramble the description of the polynomial, choose an invertible linear transformation of variables $\underline{x} = A\underline{s} \pmod n$ where A is a secret matrix of constants. When the s_i variables are substituted into P', a new d-degree polynomial $P(s_1,\ldots,s_v)$ is created which is no longer linear in any of the variables. The equation $P(s_1,\ldots,s_v) \equiv 0 \pmod n$ is apparently difficult, but can be easily solved by changing the $\underline{x}$ solutions into $\underline{s}$ solutions via the secret transformation A.

Equation (1) and solution (8) are a special case of this general technique. The P' polynomial is $x_1 \cdot x_2 - m$, and it is linear in x_1 (its linearity in x_2 as well is a coincidence). The most general solution of $P'(x_1,x_2) = 0 \pmod n$ is $x_1 = m/x_2 \pmod n$ where x_2 ranges over $\mathbb{Z}_n^*$. P' is scrambled via the linear transformation $x_1 = s_1 + s_2/u \pmod n$, $x_2 = s_1 - s_2/u \pmod n$ where u is a secret constant. In terms of s_1 and s_2 the polynomial equation becomes $s_1^2 + (-1/u^2)s_2^2 - m = 0 \pmod n$, which is exactly equation (1). This equation is quadratic both in s_1 and in s_2, but its most general solution can be easily obtained via the inverse linear transformation $s_1 = (x_1 + x_2)/2 \pmod n$, $s_2 = (x_1 - x_2)(u/2) \pmod n$ by substituting $x_1 = m/x_2 \pmod n$. The result is exactly solution (8), with x_2 playing the role of the free parameter r.

The security of this technique depends on the choice of P' and A, and each instance should be analysed carefully to verify that the transformation A scrambles the easy structure of P' sufficiently well.

ACKNOWLEDGEMENT

During the hike of the 1983 Oberwolfach meeting on Complexity Theory, H.W.Lenstra, Jr. has pointed out to us that the role of k and m in the equation $x_1^2 + kx_2^2 = m \pmod n$ can be interchanged.

References

1. Canfield, E.R., Erdös, P. and Pomerance, C.: On a Problem of Oppenheim Concerning "Factorisatio Numerorum". J. Number Theory 17 (1983), 1-28
2. Diffie, W. and Hellman, M.: New Directions in Cryptography. IEEE, IT-22 (1976), 644-654.
3. Gauß, C.F.: Disquisitiones Arithmeticae. Leipzig 1801. German translation: Untersuchungen über höhere Mathematik. Springer, Berlin 1889.
4. Lenstra, H.W. Jr.: On the Calculation of Regulators and Class Numbers of Quadratic fields. Journées Arithmétiques 1980, J. V. Armitage (ed), Cambridge University Press 1982, 123-150.
5. Ong, H. and Schnorr, C.P.: Signatures Through Approximate Representations by Quadratic Forms. Advances in Cryptology: Proceedings of Crypto 83 Plenum Publ. Corp. New York 1984, 117-132.
6. Pollard, J.M.: A Monte Carlo Method for Factorisation, BIT 15 (1975), 331-334.
7. Pomerance, C.: Analysis and Comparison of some Integer Factoring Algorithms. Computational methods in Number Theory. R. Tijdemen, H. Lenstra (Eds). Series Mathematical Centrum 154, 155, Amsterdam (1982).
8. Rabin, M.O.: Probabilistic Algorithms in Finite Fields. Siam J. on Computing 9 (1980), 273-280.
9. Rivest, R.L., Shamir, A., and Adleman, L.: A Method for Obtaining Digital Signatures and Public Key Cryptosystems. Comm. ACM 21 (1978) 120-126.
10. Schnorr, C.P. and Lenstra, H.W., Jr.: A Monte Carlo Factoring Algorithm with Linear Storage. To appear in Mathematics of Computation.
11. Schnorr, C.P. and Seysen, M.: An Improved Composition Algorithm. Preprint, University Frankfurt, 1983, submitted for publication.