NTIA REPORT 81-79

A Critique of Some Public-Key Cryptosystems

W.J. Hartman

U.S. DEPARTMENT OF COMMERCE
Malcolm Baldrige, Secretary

Bernard J. Wunder, Jr., Assistant Secretary
for Communications and Information

August 1981

TABLE OF CONTENTS

LIST OF TABLES

LIST OF SYMBOLS

$d|x$: d divides x means, if d and x are integers, there exists an integer k such that $k \cdot d = x$.

$\gcd(x,y)$: greatest common divisor d of x and y means $d|x$ and $d|y$, and if $k|x$ and $k|y$, then $k|d$.

$[x]$: The greatest integer in x, also called the integer part of x or int (X).

$A \pmod b$, or $X \equiv A \pmod b$: means that X is such that $0 \leq X < b$ and $b|(X-A)$.

$(a|P)$ is the Jacobi Symbol which, for odd P has the following properties:

1. $(a|P)\ (b|P) = (ab|P)$
2. if $a \equiv b \pmod P$, $(a|P) - (b|P)$
3. if $\gcd(r,P = 1$, $(ar^2|P) - (a|P))$
4. $(-1|P) - (1-1)^{\frac{P-1}{2}}$
5. $(2|P) - (-1)^{(P^2-1)/8}$
6. $(P|Q)\ (Q|P) = (-1)^{(P-1)/2 \cdot (Q-1)/2}$
   (for P, Q both odd).

$O(n)$: order of N means, $x = O(n)$ implies that $x < c \cdot n$ for some constant c as n approaches a limit. In this paper the limit will be ∞. Similarly $x = O(f(n))$ means that $x < c \cdot f(n)$ as n approaches a limit.

$Q(n)$: Euler's totient function. If $n = P_1^{k_1} P_2^{k_2} \quad P_k^{2k}$ is the decomposition of n into its distinct prime factors, $Q(n) = n(1-\frac{1}{P_1})(1-\frac{1}{P_2})\ldots(1-\frac{1}{P_2})$.

A CRITIQUE OF SOME PUBLIC-KEY CRYPTOSYSTEMS

W.J. Hartman*

Several of the public-key cryptosystems that received considerable publicity are examined. The uses, implementation, and potential cryptoanalytic attacks are discussed. Since most of the suggested systems depend on the manipulation of large numbers, special multiprecision computer programs have been developed to demonstrate system implementation and cryptoanalytic attacks. Examples of the use and timing of these programs are included.

Key words: Public-key; number theory; cryptology; algorithms

1. INTRODUCTION

The opportunities and benefits provided by technological developments in electronics and computers are being increasingly exploited for the transfer and processing of information. However these innovative applications also present a challenge to the user to identify and correct undesirable side effects.

Although a clerk can access files stored in a computer more efficiently than those stored in a filing cabinet, he can visually monitor those who gain access to the filing cabinet but not those who gain access to the computer.

A company sending proprietary information across country by mail would know if the information had been stolen, but one sending information by radio has no way of knowing if the information is obtained by someone else.

These are examples of problems which we will classify as protection problems. In this paper, we examine the role that a special class of cryptographic systems plays in the solution of these problems.

The historical use of cryptography was to protect against information's being obtained by unauthorized persons. This will be called message protection. More recent applications of cryptography include electronic signatures, verification, and protection against message alteration.

An electronic signature is a type of message which serves the same purpose as an ordinary signature, i.e., it can be generated by only one person, it uniquely identifies that person, and it can be substantiated legally. See e.g., Rabin (1978), and Popek and Kline (1978).

*The author is with the Institute for Telecommunication Sciences, National Telecommunications and Information Administration, U.S. Department of Commerce Boulder, CO 80303

Authentication is similar to signatures, but may identify a group instead of an individual, and may be only for one-time or short-term use.

Prevention of message alteration is a scheme which may or may not conceal the message, but assures that no alteration can take place between the originator and the intended recipient. The introduction of the idea of Public-key cryptosystems (PKC) by Diffie and Hellman (1976) was directed at the solution of some of these non-commentioral cryptologic problems. The idea gained impetus with the publication by Rivest et al (1978) of an algorithm for a PKC which appears quite secure. This algorithm will be designated the RSA-PKC.

What is a PKC?

In a conventional cryptosystem both the sender and recipient of a message must have a common secret, usually in the form of a key, in order to communicate. In a PKC, only the recipient holds the secret (key). This is accomplished through the use of either one-way or trap-door functions which have the property that, given the function f, it is easy to calculate the values

$$y = f(x),$$

but it is difficult, or impossible to calculate x given y. In the case of the trap-door functions, knowing the secret key also makes it easy to calculate values of $x = f^{-1}(y)$.

In order to examine the concepts of PKC in more detail, it is necessary to establish some notion of computational difficulty. One concept is based on key length (N bits) and the number of operations a cryptanalyst needs to perform to obtain the key. If the cryptanalyst requires approximately 2^{N-1} operations we define this as exhaustive search. Throughout, an operation may be a simple arithmetic operation such as a multiplication or a complex operation such as a complete message encryption. The interpretation should be clear in context.

Similarly, we define

	Required Operations	Name
	a^N, $1 < a < 2$	Exponential
	$N^{f(N)}$, $f(N)$ increases with N,	Mixed
and	$f(N) < \frac{N}{\log_a N}$	Mixed
	N^k, $k > 1$, k fixed	Polynomial

in decreasing order of cryptanalytic difficulty.

A second concept is based on the relationship between the number of operations (N) required by the legitimate recipient of a message to decrypt and the number of operations required by the cryptanalyst to decrypt the message. Within this framework it is not necessary that the cryptanalyst obtain the key. The difficulty is then designated as follows:

Required Operations	Name
a^N, $a > 1$	Exponential
$N^{f(N)}$, $f(N)$ increases with N, and $f(N) < \frac{N}{\log_a N}$ for some $a > 1$.	Mixed
N^k, $k > 1$, k fixed	Polynomial

Some examples follow:

(a) It is estimated that it would require approximately 2^{54} (approximately 1.8×10^{16}) operations on the average to find the 56 bit DES (Data Encryption Standard, 1977) key given both plain text and cipher text. Assuming one operation each nanosecond (ns), (10^{-9}s), finding one key would take 1.8×10^7 seconds, or 0.6 years. This rate is far above available equipment capabilities which only are in the range of of one operation every 100 ns. At this speed, it would require 60 years to find the key.

On the other hand, if polynomial time (say N^3) were required, it would take 1.7×10^5 operations, which even at today's speed would take less than 1 second.

(b) The second concept, comparing the work required by the legitimate recipient and the cryptanalyst is illustrated by Merkle's (1978) puzzle PKC. It is estimated that, if it requires N units of work by the legitimate recipient (LR), it requires N^2 units by the cryptanalyst (C). Thus, although N could be made large enough so that it would be unfeasible for (C) to recover the message in a reasonable time, the work for (LR) would become burdensome. Consequently, although Merkle's (1978) scheme may have some specialized uses, it will not be considered further in this paper.

Further examples of these comparisons are given later.

Overhead and Message Expansion

In what follows, we will assume that the messages are either all numbers (such as transmitting keying material) or mixed numbers and letters in which case we will assume a 64 character set. This may not be efficient in terms of overhead, but is most convenient for computer implementation. Thus, if one converts each character of the set into two decimal digits, a message expansion of 100/64 occurs. On the otherhand, if one does not convert each character to a decimal marker, but instead uses strings of the binary representating characters as a single binary number which is less than M, one obtains a message expansion of $M/_2k$ where 2^k is the largest power of 2 less than M. For large M, this can be very close to 1. If, in addition to this, the cryptographic system maps a subset, S, of numbers less than M into a subset of the numbers less than N, N > M, the message expansion becomes N/S(M) where S(M) is the number of elements in S.

The overhead is the amount of work required to establish communications, set up the cryptosystem, maintain synchronization, etc. The only overhead considered here will deal with obtaining the necessary parameters to establish a PKC.

2. THE LU AND LEE PKC

Lu and Lee (1979) proposed a PKC based on number theoretic concepts. The system was immediately shown to be unacceptable from a security standpoint. It is presented here as an example of a system for which the cryptanalyst need not find the secret key (and, in fact, may not be able to do so) but can still read the plain text with only a small increase in work over legitimate decryption.

The public keys are integers C_1, C_2 and R, with R the product of two primes P_1 and P_2, and C_1 and C_2 both relatively prime to R with $C_1 + C_2 > R$.

The encryption process takes two message segments m_1 and m_2 and forms the encrypted message X from

$$X = (C_1m_1 + C_2m_2) \bmod R. \tag{1}$$

The decryption process uses the (secret) knowledge of P_1 and P_2. The four parameters

$$a_{11} \equiv C_1 \bmod P_1,\ a_{12} \equiv C_2 \bmod P_1 \tag{2}$$

$$a_{21} \equiv C_1 \bmod P_2,\ a_{22} \equiv C_2 \bmod P_2 \tag{3}$$

are calculated.

The messages are then obtained by forming $X_1 \equiv X \bmod P_1$, $X_2 \equiv X \bmod P_2$, and calculating

$$m_1 = \frac{X_1 a_{22} - X_1 a_{21}}{a_{11}a_{22} - a_{12}a_{21}} \tag{4}$$

and

$$m_2 = \frac{X_2 a_{11} - X_1 a_{21}}{a_{11}a_{22} - a_{12}a_{21}} \quad . \tag{5}$$

To insure unique decryption, it is necessary to limit the size of the messages by publishing two numbers,

$$M_1 \leq \frac{1}{2} \min \left[\frac{q}{a_{12}} , \frac{q}{a_{21}} \right]$$

$$M_2 \leq \frac{1}{2} \min \left[\frac{q}{a_{12}} , \frac{q}{a_{22}} \right]$$

where $q = \min (P_1, P_2)$, and $[y]$ denotes the greatest integer in y.

The first to point out the weakness of this PKC were L. Adleman and R. Rivest of the Massachusetts Institute of Technology who had considered the system and rejected it. Others who have examined the weaknesses were L.N. Lee (1979), C. Osgood (1979), and M.J. Kochanski (1980).

The observation central to all of the methods of cryptanalysis is that, because of the bounds M_1 and M_2 on the message size, it is possible to reduce the problem to one of diophantine analysis.

Since g.c.d. $(C_1, R) = 1$, $C_1^{-1} (\bmod R)$ can be calculated, and

$$m_1 = (C_1^{-1} x - C_1^{-1} C_2 m_2) \bmod R$$

defines m_1 in terms of a linear congruence relation in m_2, which has a unique solution in the range $0 \leq m_1 \leq M_1$, $0 \leq m_2 \leq M_2$.

We first note that the a_{ij} must be large enough to make the search for k such that

$$\text{g.c.d.}(C_i - k, R) > 1$$

is impossible, since $\text{g.c.d.}(C_1 - a_{11}, R) = P_1$

$$\text{and g.c.d.}(C_1 - a_{21}, R) = P_2$$

for example.

However, making the a_{ij} large decreases the M_i with a resultant increase in message expansion. Thus, with the primes about 100 digits and the a_{ij} about 15 digits a 17% expansion occurs.

Kochanski (1980) suggests a linear change of variable from C_1, C_2 into C_1', C_2', where C_1' and C_2' are such that

$$C_1'M_1 + C_2'M_2 < R$$

in which case the solution $\{M_1', M_2'\}$ is easily obtained, from which the solution to the original problem can be calculated.

Osgood (1979) defines b, $0 \le b \le r$ as $b \equiv C_2^{-1}x \pmod R$, where x is the encrypted message. He defines $d = -C_1C_2^{-1} \pmod R$ and notes that the original messages m_1 and m_2 must be in the set $(\ell, (b + \ell d) \bmod R)$ with $0 \le \ell \le M_1$ and $0 \le (b + \ell d) \bmod R \le M_2$ and, by the uniqueness, there is only one pair $(\ell, (b + \ell d) \bmod R)$ in this range. Let $||\ frc(X)\ ||$ be the smaller of frc(X) and 1 - frc(X), where frc(X) is the fractional part of X. Then the equation to be solved is

$$|| \frac{b}{R} + \ell \frac{d}{R} || = \frac{m_2}{R} .$$

This is a nonhomogeneous problem in diophantine approximation. He then approximates d/R by continued fractions and shows that for almost all d/R a solution to the equation is produced, which in turn produces the messages. Both of these methods of cryptanalysis can be implemented to run with only a small amount of increase in work over that required of the legitimate recipient.

Modifications of this method have been proposed, such as using three primes and three coefficients, but these also appear to have weaknesses.

The general consensus is that this class of PKC's is at best marginal because of the linearity and other structure present.

3. THE KNAPSACK PKC

Merkle and Hellman (1978) have used the knapsack problem from combinatorics as a basis for a PKC. The general knapsack problem is known to belong to class of hard problems called NP-complete, Kays (1972), but the cryptographic knapsack is not known to belong to this class.

The Algorithm

Merkle and Hellman (1978) describe an additive and a multiplicative knapsack PKC. We restrict our attention here to the additive algorithm.

The user A generates a vector $\{a_i\}$ such that

$$a_{i+1} > \sum_{j=1}^{i} a_j, \quad i = 1,2,\ldots,n.$$

He then chooses $m > \sum_{j=1}^{n} a_i$ and W with g.c.d. (W,m) = 1 so that w^{-1} mod m exists. He then forms $b_i \equiv wa_i \pmod{m}$ and checks to see that $b_i \neq \sum_{j \neq i} b_j$ for i = 1,2,...,n. If this condition is met, he publishes the vector $\{b_i\}$. He keeps secret $\{a_i\}$, w, and m. Anyone wishing to send a message to A forms the binary message vector $x = \{x_i\}$, i = 1,2,...,n, (x_i = 0 or 1), calculates $y = \sum_{1}^{m} x_i b_i$ and sends y to A. Then A calculates $z = w^{-1}y \pmod{m}$ from which he is then able to calculate x as follows: Let $C_i <| D$ mean C_i is the largest element of the set $\{C_i\}$ which is less than D.

Initialize:

$\{x_k\} = \{0\}$,

$z = z_0$

repeat:

set $x_k = 1$ if $a_k <| z$

and set $z = z - a_k$,

$(i = i+1)$

until: z = 0.

This terminates after at most n steps with the message vector x.

Cryptanalysis

It is easily seen that exhaustive search requires approximately 2^{n-1} tests. The best known algorithm requires $O(2^{n/2})$ steps. Thus n must be sufficiently large to preclude this attack.

This PKC is susceptible to the "partially known" plain text attack. If k of the n bits are kept constant, where n-k is small, finding the additional n-k bits would be quite easy. As an example, let n = 200, and suppose k = 180 corresponding to a 30-letter signature. Then at most 2^{10} combinations of the remaining bits would need to be checked to get the entire message. Even if the k bits were shifted in the message, an additional work factor of (n-k) would be all that was required for decryption. Of course, this does not provide decryption for other messages.

Herlestrom (1978) has proposed a more general type of attack. He suggests that the public key set $\{b_i\}$ be transformed by choosing $B > \Sigma b_i$ and g.c.d.(B, b_i) = 1 and forming $c_{ij} \equiv b_j^{-1} b_i \pmod{B}$, i = 1,2,...,n; j = 1,2,...,n.

If for some j, there exists an $i = \alpha$, such that

$$C_{\alpha j} > \sum_{i \neq \alpha} C_{ij} ,$$

then the transformed encrypted message

$$b_j^{-1} y (\text{mod } B)$$

is reduced to n-1 components. A similar transformation is performed on the remaining n-1 coefficients, and the process continued until all are reduced. He claims success in producing such a $C_\alpha j$ in a small number of B selections. However, no upper limit is established on the number of trials before a suitable B is found. If the process of generating $\{C_i\}$ from $\{b_i\}$ is random, the probability of finding a suitable set $\{C_i\}$ is $1/(n-2)!$ at most. Thus, until the method is tested over large sets (n>200), its effectiveness is dubious.

Another method which has been suggested is to pick numbers P_i, $i = 1,2,\ldots n+k$ and w_i for $i = 1,2,\ldots,n+k$, $P_1 = 1$, $w_1 = 1$ and form n+k equations in n unknowns as follows:

$$\text{Let } C_{ij} = \text{int } \left(\frac{w_j b_i}{P_j}\right)$$

$$\text{and} \qquad y_j = \text{int } \left(\frac{w_j y}{P_j}\right), \; j = 1,2,\ldots,n+k.$$

Pick the equation with j = 1, and any n-1 equations from the n+k-1 remaining equations and solve. For small n the method produces frequent solutions. Additional testing for larger n is necessary before the feasibility can be established. In any case, the method greatly increases the work factor.

Implementation

Merkle and Hellman (1978) suggest the following implementation for n = 100. For larger n, the extension is clear. Choose a_i randomly from the interval

$$[(2^{i-1}-1)\cdot 2^{100}+1, \; 2^{i-1}\cdot 2^{100}].$$

Choose m in the interval $(2^{201}+1, \; 2^{202}+1)$ and w in the interval (2, m-2). Check to insure that g.c.d. (w,m) = 1. Using these values, there is a slightly greater than 2 to 1 message expansion.

When the set $b_i \equiv wa_i \pmod m$ is formed, it should be checked for possible weakness such as having one number larger than the sum of the others, or having possible geometric progressions between elements. Note that the published key for this example has approximately 6100 decimal digits.

4. THE RSA PKC

In contrast to the Lu and Lee system, the RSA system has withstood attempts at systematic cryptanalysis. However, the work toward breaking the system has been fragmented leaving the question of the security of the system open. In the later parts of this section we give suggested methods of attack and, in some cases, the probability of success of the attack.

The System

If two users, A and B, wish to communicate using the RSA algorithm they would do the following.

User A would choose two large primes p_A and q_A and form $r_A = p_A q_A$. He would then form $\phi(r_A) = (p_A-1)\ (q_A-1)$ and choose s_A such that $(s_A, \phi(m_A)) = 1$. He would then find t_A such that $s_A t_A \equiv 1 \pmod{\phi(m_A)}$. This is done using the extended version of Euclid's algorithm (Knuth, 1969) which finds x and y such that

$$x2 + y\ Q\ (m) = 1.$$

If x is positive, $t_A = x$, otherwise

$$t_A = Q\ (m) + x.$$

He would then tell B the values r_A and s_A.

Similarly, B would choose two large primes p_B, q_B and tell A the values r_B and s_B.

A would block his message into groups of numbers X_{Ai} with $X_{Ai} < r_B$ and transmit

$$X_{Ai}^{s_B} \equiv Y_{Ai} \pmod{r_B}.$$

B would receive Y_{Ai} and calculate $Y_{Ai}^{t_B} = X_{Ai} \pmod{r_B}$ to obtain the message set X_{Ai}. Similarly, B would transmit

$$X_{Bi}^{s_A} \equiv Y_{Bi} \pmod{r_A}$$

to A, and A would recover the message X_{Bi}, calculating

$$Y_{Bi}^{t_A} \equiv X_{Bi} \pmod{r_B}.$$

It is important for A to keep p_A, q_A and t_A secret and for B to keep p_B, q_B and t_B secret.

Since the encrypted messages and the messages are the same length ($<r$), it is apparent that only a small amount of message expansion is involved in this use of the algorithm.

Implementing the Algorithm

The first step in implementing the algorithm is to find two primes p_1 and p_2. The size of the primes determines the relative security of the algorithm, and for minimal security 50 digit primes are suggested. Other restrictions on these primes will be added in later sections discussing the cryptanalysis. The second step is to select the encryption exponent, s, where s is relatively prime to P_1-1 and P_2-1 and sufficiently large so that $2^s > P_1P_2$. Although any s satisfying these two conditions is adequate, selecting s to be of the form $2^k + 1$ allows very efficient encryption, (k + 1) multiplication and modulus reductions. For any $s < P_1P_2$, at most $2\{\log_2 P_1P_2\} + 1$ multiplication and reductions are needed. Additional restrictions may be placed on s as will be seen later.

Since all of the arithmetic operations are multiple precision, it is necessary to design efficient programs for the basic operations of addition, subtraction, multiplication and division. Such routines have been developed under contract for the Institute for Telecommunications Sciences (ITS) for use on CDC-6600 series[1], or CDC-750 series computers. Table 1 gives approximate timing for the operations on the CDC-6600.

[1] "Certain commercial equipment, instruments, or materials are identified in this paper to specify adequately the experimental procedure. In no case does such identification imply recommendation or endorsement by the National Telecommunications and Information Administration, nor does it imply that the material or equipment identified is necessarily the best available for the purpose."

Table 1.

Operations/Second

Decimal Digits	ADD	SUB	MULT	DIV
200	22,000	21,000	3,600	1,200
1000	8,500	7,600	180	75

Thus, with a 200-decimal-digit modulus, it will be possible to encode or decode one block (200 digits) in less than 2s. See Appendix B for some actual timings. This equates to approximately 330 bits/s, which is slow for many applications. As will be seen, certain applications will require double encryption and decryption entailing further reductions in communication rates.

The same precautions that are observed when using a conventional cryptosystem should be observed when using a PKC. For example, if the RSA-PKC is used in the block mode, a few random bits should be used to pad each message segment to avoid the possibility that an often-repeated segment can be obtained in plain text through indirect means.

Two methods of distributing key systems for PKC's have been proposed. The first, a published directory, and the second, by key exchange through the information transmission medium. If active wiretapping is a possibility, the second method negates any security inherent in the PKC. One possible solution to this problem is to have a key verification center (Michelman, 1979). In what follows, we will not consider this problem further, restricting our attention to passive intercept.

Using the PKC's

One of the most important applications of PKC's is for the distribution of conventional keying material. This is so because of the combination of the desirable feature of very high encryption and decryption rates achieved by the conventional systems as compared to the PKC's and the undesirable problems associated with the key management for the conventional systems. User A wishes to communicate with User B using DES with no prior DES key exchange. User A calls user B and, using user B's public key set, encrypts the information concerning the use of the DES. This information would include a randomly generated number of at least 64 bits for key use, and might contain information on the mode of DES operation such as cipher-feedback, etc., and a seed variable, if applicable.

Users A and B are then able to use DES encryption for the further exchange of information.

For protecting messages against possible changes and for providing some authentication, the following procedures are recommended. Users A and B determine which keying variable r_A or r_B is smaller, say, r_A. User A sends message X to user B by first encrypting X using his secret exponent t_A to form $X^{t_A} \equiv Z \pmod{r_A}$ and then forming $Z^{s_B} \equiv Y \pmod{r_B}$, a double encryption. User B decrypts using first $Y^{t_B} \equiv Z \pmod{r_B}$ and then $Z^{s_A} \equiv X \pmod{r_A}$. As long as user A's secret keys are known only to him, he would be the only one who could send this message, if some precautions, such as numbering the messages are observed. Note that someone intercepting the message could retransmit it later and have it accepted as genuine if duplicate messages were allowed.

This scheme has been proposed by Simmons (1979) for use in transmitting information from sensors used to monitor nuclear detonations. These sensors would be installed in a foreign country, and the scheme would be used to insure that the foreign country was not sending spurious information, while at the same time allowing the foreign country to monitor the transmissions to insure that the transmissions were not being used to convey information of a different nature.

Although at present it is thought extremely unlikely that the secret keys can be obtained by cryptanalysis, the minute chance of success casts some doubt on the acceptance of the above authentication method, even with modifications, as a method for digital signatures. The fact that digital signatures, like actual signatures, must be able to withstand legal tests as well as expert scrutiny for long periods into the future makes quick adaptation unlikely, except for special purposes.

For certain applications such as encrypting data for storage on tapes, where there is no need to disclose any of the keying variables, much smaller values can be used for the primes, allowing more rapid transmission rates. In general, this application would be most appropriate for individuals or organizations with infrequent encryption requirements.

Cryptanalysis

The most obvious method of cryptanalysis of an RSA-PKC is factoring the public key component r. The knowledge that factoring has long been considered a difficult problem was one of the considerations leading to the construction of

the RSA-PKC. Appendix A gives an overview of factoring methods. It is shown there that choosing primes p such that $p \pm$ have many small factors leads to more rapid factoring algorithms. Consequently, in choosing the primes for the RSA-PKC, care must be exercised. Later in this chapter a method for choosing primes is detailed.

The two primes chosen should be sufficiently different so that the differences-of-squares method of factoring does not work quickly.

With these precautions in selecting primes, none of the methods of factoring $r = p_1 p_2$ given in Appendix A appears feasible for factoring an r of 200 decimal digits. However, as Guy (1975) points out, factoring large numbers is like gambling, with the odds of success extremely small, but, with a lucky guess, such as choosing the "right" k in the Morrison and Brillhart (1975) method, success can be achieved by the cryptanalyst.

A second type of attack has been suggested by Simmons and Norris (1977) with extensions by Herlestram (1978). The attack is as follows: For $1 < x < r-1$, define

$$E_s^1(x) = x^s \bmod r = y_1$$

$$E_s^2(x) = (x^s)^s \bmod r = y_1^s \bmod r = y_2$$

$$\vdots$$

$$E_s^n(x) = x^{s^n} \bmod r$$

and for any polynomial $P(t) = \sum_{i=0}^{k} a_i t^i$ we have $X^{P(s)} = \prod_{i=1}^{k} E_s^i(x^{a_i}) \pmod r$. Now, if for some encryption exponent g, $x^g \equiv y \bmod r$ and some polynomial P(t), $y^{p(g)} \equiv y \pmod r$, then we can find the original message, and in fact, with the suggested selection of primes, we can factor r (Blakley and Blakley, 1978; 1979(a); 1979(b); Williams and Schmid, 1979). However, the selected primes insure that the probability of a decryption by this method is extremely small.

The cryptanalyst knowing r and s has the potential for cataloging plain text-ciphertext pairs within practical storage limitations. Thus, if the message set is limited, he has the potential for decrypting many of these messages without attempting factoring or the other analysis methods discussed above. However, few restrictions are placed on the choice of the encryption exponent s, and the calculation of the decrypting exponent t involves about the same amount of work as

encrypting or decrypting a message making it possible to change s frequently. This would require the recalculation of the entire catalogue that the cryptoanalyst had stored each time s was changed.

Each of the above methods has a small chance of succeeding in a reasonable amount of time. However, on the whole, the estimate of Rivest, et al (1978) of the time required, 3.8×10^9 years, to factor N of 200 digits, appears reasonable within a factor of 10 or 10^2, which clearly places it in the unfeasible category.

Detailed Implementation of the Algorithm

In order to reduce the probability of success of the methods of attack formulated in the last section it is desirable to have a prime p such that $p = 2p'+1$ where p' is prime and such that $p + 1$ has a large prime factor. First select two random primes, α, β, of about 50 digits. Find ϕ such that

$$\phi\beta \equiv -1 \pmod{\alpha} \quad 0 < \phi < \alpha, \text{ and form}$$

$$k(2\alpha\beta) + 2\phi\beta + 1 = x(k) \text{ and}$$

$$k(4\alpha\beta) + 4\phi\beta + 3 = x_2(k) = 2x_1 + 1$$

and increase k until x_1 and x_2 are both primes, $x_1(k) = P'$ and $x_2(k) = P$. If no such primes are found before k reaches a previously set limit, L, return to the first step and find more (one or two) primes and repeat.

To test whether $x_1(k)$ and $x_2(k)$ are primes, first sieve the pairs $x_1(k)$ and $x_2(k)$ using the 167 odd primes less than 1000. For the remaining pairs find those that satisfy

$$3^{x_1(k)-1} \equiv 1 \pmod{x_1(k)}$$

and

$$3^{x_1(k)} \equiv 1 \pmod{x_2(k)}.$$

It is then sufficient to prove $x_1(k)$ prime. Since $x_1(k) - 1 = 2\beta(k\alpha+w)$, we need only to factor $k\alpha+w$, or prove it prime using the methods in Appendix A. Note that $k\alpha+w$ is approximately 50 digits.

We have placed only limited restrictions on the choice of the encryption exponent. Williams and Schmid (1979) give a method for calculating a "good" exponent, but the selection of the primes makes the probability of finding a "bad" exponent extremely small so we shall not present the details here.

When encoding or decoding, the input data need to be arranged into blocks. If only numerical information is transmitted, the block length can be either the

length of the number string or the length of r. For the 64 character set, it may be convenient to block in 90-character strings corresponding to a typical typed line length. This would be 540 bits or 163 digits, resulting in a message expansion of about 23%. The user must determine the convenience and economy tradeoffs.

The calculations of $m^s \pmod r$ are done (using multiple precision operations) using the standard power raising techniques given in Rivest, et al (1978) and Knuth (1969) among others.

5. REFERENCES

Blakley, B., and G.R. Blakley (1978), Security of number theoretic public key cryptosystems against random attack, I, Cryptologia 2(4), pp. 305-321.

Blakley, B., and G.R. Blakley (1979a), Security of number theoretic public key cryptosystems against random attack, II, Cryptologia 3(1), pp. 29-42.

Blakley, B., and G.R. Blakley (1979b), Security of number theoretic public key cryptosystems against random attack, III, Cryptologia 3(2), pp. 105-118.

Data Encryption Standard, (1977), National Bureau of Standards, Federal Information Processing Standard Publication No. 46.

Diffie, W., and M.E. Hellman (1976), New directions in cryptography, IEEE Trans. Inform. Theory, Vol. IT-22, Nov., pp. 644-654.

Guy, R.K. (1975), How to factor a number, Congressus Numerantium XVI, Proceedings Fifth Manitoba Conference on Numerical Mathematics, Winnipeg, pp. 49-89.

Herlestrom, Tore (1978), Critical remarks on some public-key cryptosystems, BIT, 18, pp. 493-496.

Karp, R.M., (1972), Reducibility among combinatorial problems, in complexity of computer computations, R.E. Miller and J.W. Thatcher (Eds.), New York, Plenum, pp. 85-104.

Knuth, D.E. (1969), The Art of Computer Programming, Vol. II, Seminumerical Algorithm, Addison-Wesley (Reading, MA).

Kochanski, M.J. (1980), Remarks on Lu and Lee's proposal for a public-key cryptosystem, Cryptologic, Vol. 4, #4, October, pp. 204-207.

Lee, L.N. (1979), Note on cryptosystems Comsat Technical Review, Vol. 9, #2B, Fall, pp. 717-721.

Lu, S.C., and L.N. Lee, (1979), A simple and effective Public-key cryptosystem, Comsat Technical Review, Vol. 9, #1, pp. 15-24.

Merkle, R. (1978), Secure communication over insecure channels, Commun. ACM, Apr., pp. 294-299.

Merkle, R., and M.E. Hellman (1978), Hiding information and signatures in trap door knapsacks, IEEE Trans. Inform. Theory, Vol. IT-24, Sept., pp. 525-530.

Michelman, E.H. (1979), The design and operation of public-key cryptosystems, National Computer Conference, pp. 305-310.

Morrison, M.A., and J. Brillhart (1975), A method of factoring and the factorization of F_7, Math. Comp., 29, pp. 183-205.

Osgood, C.F. (1979), Breaking a public key code using diophantine approximation, Presented at the 1979 West Coast Number Theory Conference.

Popek, G.J., and C.S. Kline (1978), Encryption protocols, public-key algorithms. and digital signatures in computer networks, Foundations of Secure Computation, Section II, pp. 133-153.

Rabin, M.O. (1978), Digitalized signatures, Foundations of Secure Computation, Section II, Academic Press, Inc., pp. 155-168.

Rivest, R.L., A. Shamir, and L. Adleman (1978), On digital signatures and public key cryptosystems, Commun. ACM, Vol. 21, No. 2, Feb., pp. 120-126.

Simmons, G.J. (1979), Message authentication without secrecy: A secure communications problem uniquely solvable by asymmetric encryption techniques, IEEE Publication 79CH 1476-1AES.

Simmons, G.J., and M.J. Norris (1977), Preliminary comments on the MIT public key cryptosystems, Cryptologia, Vol. I, Oct., pp. 406-414.

Solovay, R. and V. Srassen (1977), A fast Monte Carlo test for primality, SIAM J. of Computing, 6, pp. 84-85.

Williams, H.C., and B. Schmid (1979), Some remarks concerning the M.I.T. public-key crypto systems, BIT 19, pp. 525-538.

APPENDIX A. FACTORING

R.K. Guy (1975) has written an excellent summary of factoring methods. In this appendix we concentrate on a few of these that appear most appropriate for our application.

Traditionally the most difficult factoring problem is one where the number to be factored has only a few large prime factors. In the present case we are most concerned with numbers which are the product of two distinct prime factors, even though other cases arise in proving a number to be prime.

Most factoring methods begin with instructions to test first whether any primes $P_i<b$, where b is some preselected factorbound, divide the number N to be factored. Since we know that N is a product of two large primes, this step is eliminated. However, for other purposes, a factor base is useful, and should be available. For a small bound on the factors, a simple sieve determined by a set of arithmetic progressions is useful. The general form is given by (Lehmer, 1953) the set of numbers N satisfying all of the relations

$$P_{ij}(x) = m_i x + a_{ij} \quad \left\{ \begin{matrix} i = 1,2,\ldots k \\ j = 1,2,\ldots n_i \end{matrix} \right\}$$

where the m_i are relatively prime in pairs. One very useful case is sieving out multiples of 2 and 3 by examining only those numbers of the form $6k\pm1$. As an example, this gives a set of 333 numbers less than 1000. Approximately half of this set (167) are odd primes. Large numbers of such relations lead to very complicated functions for programming on a general-purpose computer although efficient special purpose equipment (Lehmer, 1966) has been constructed.

Squares play an important role in many factoring methods. If one can find two numbers x and y with $x^2-y^2=N$ and $x-y\neq1$, then $N=(x-y)(x+y)$ and N is factored. Similarly, if

$$\left.\begin{matrix} x^2 \equiv y^2 \pmod N \\ \text{and} \quad 1 < \gcd(x-y,N) < N, \end{matrix}\right\} \quad (**)$$

then N is factored. Thus, many of the methods are designed to find x and y satisfying (**).

The simplest method is to begin with $[\sqrt{N}]+1=A$, and form $[A^2]-N=B$ and increase B by consecutive odd increments $(2A+2k+1)$ until B is a perfect square. In our case, $N=pq$, $q>p$ and the number of steps required is $O(p)$ which is not feasible. Even with sieving and computational shortcuts (Guy, 1975), it is not suitable for p, q~100 digits. The use of multipliers does offer the possibility of some

improvement. If q is approximately kP, or more generally if $\ell q \sim kp$, then $k\ell N$ may be rapidly factored into the factors kP and ℓq. Specifically, let i=1,2,... and j=1,2,... $[\log_2 N]$. Form iN and check if $([\sqrt{iN}]+j)^2 - iN$ is a perfect square. If [q/p] is not too large, we might expect success with reasonable sized i. However, one can estimate the size of the i needed by considering the k^{th} convergent, A_k/B_k in the continued fraction expansion of q/p; then $i = A_k \cdot B_k$ for some k, and i is probably large. The best systematic attack of this type is described by Lehman (1974) and requires $O(N^{1/3})$ steps.

A number of methods have developed around the idea of finding a sufficient number of quadratic residues so that a perfect square can be formed. Thus, one finds

$$x_i^2 \equiv a_i \pmod{N}$$

until some set s of a_i is such that

$$\prod_{a_i \in s} a_i = B^2 \text{ and}$$

$$1 < \gcd\left(\prod_{a_i \in s} x_i - B,\ N\right) < N$$

Morrison and Brillhart (1975) use the continued fraction expansion for $\sqrt{kN}$ to find the x_i and a_i. They only consider a_i that have prime factors belonging to a predetermined factor base, A. The exponents for each prime in A are calculated modulo 2, forming a set of binary vectors which are used to determine the set of a_i whose product forms a square. This method is the fastest known general factoring method and runs in $O(\ell^{\sqrt{\log N \log\log N}})$ operations. The method appears to be slowest when N is the product of two primes, and it seems to make little difference in running time whether the two primes are close together or not. The choice of the multiplier k is usually made by examining a number (~5000) of Q_m for each k and choosing that k which produces the largest number of factored Q's. No a priori method of choosing k is known and a breakthrough in this area might provide a much faster factoring algorithm. A considerable amount of work has been done toward choosing appropriate factor bases, but this does not appear to be promising for obtaining large reductions in the number of required operations.

Miller (1975) proposes using a set of small primes $\{P_i\}$ i=1,2,...k and finding representations of N of the form

$$N = \Pi P_i^{\alpha_{ij}} + (1)^{e_j} \Pi P_i^{\beta_{ij}} .$$

If enough such representations can be found, a set of simultaneous equations can be solved. Let X_0 be the variable associated with -1, and X_i the variable associated
with P_i. Then the sets of equations are of the form

$$\sum_{i=1}^{k} \alpha_{ij} X_i = e_j X_0 + \sum_{i=1}^{k} \beta_{ij} X_i .$$

$$j = 1,2,\ldots,m,$$

with $m \geq k+1$. Usually a number of additional equations are needed since some of those found may not be linearly independent. Once a solution is found, N may be factored using the relation between the solution and $\phi(N)$. The biggest drawback to this method is the lack of a systematic way of finding the representations of N. Further work on this method may produce surprising results.

Pollard (1975) describes what he calls a "Monte Carlo" method. The name came from the (hoped-for) similarity of the functions to random-number generating functions. Let

$$X_{m+1} = f(X_m) \pmod N.$$

Then there exists m such that

$$X_{n+m} = X_m \text{ for all } n > n_0.$$

If p is the smallest prime dividing N and if f is "sufficiently random," then for some m, g.c.d.$(X_{2m}-X_m, N) > 1$. If g.c.d.$(X_{2m}-X_m, N) \neq N$, a divisor of N has been found. Under certain assumptions Pollard (1975) shows that $m=O(p^{1/2})$. The least number m that yields a divisor consists of two parts, a tail and a periodic part. Certain functions are known to be "bad" for this factoring algorithm since they have a maximum length period. The two functions $X_{m+1}=(X_m^2 \pm 1)$ with $X_0=2$ as a starting value have been the most thoroughly investigated. Guy (1975) gives tables showing the largest m for primes up to 10^6 and these are approximately equal to $\sqrt{p \ell n p}$. However, in this context, it should be noted that it is seldom the case that m is this large for both $f(x)=X^2+1$ and $f(x)=X^2-1$ even when $X_0=2$ is used, and for some values of X_0

it can be very small, for example if one is lucky enough to choose $X_0 \equiv \pm$ mod p or $X_0 \equiv \pm 1$ mod q. However, the probability of this happening is extremely small for large p and q. Brent (1980) has described a new cycle finding algorithm which reduces the constant in Pollard's algorithm and uses higher order polynomials for some special cases. However, the use of higher order polynomials for f(x), which offers the potential of much smaller m, has usually been rejected because of the increased number of operations required to calculate each X_i, which offsets any benefit of short cycles. The special cases Brent (1980) considers are for numbers for which it is known that the prime satisfies p=1 (mod m) and then the function $f(x)=x^m-1$ is used. For the cases we are interested in, no such relation is known, and is prohibited if the primes are chosen as suggested. Brent's cycle finding routine is

```
y := x0;
r := 1;
q := 1;
repeat
      x := y;
      for i := 1 to r do y := f(y);
      k := 0;
      repeat
           ys := y;
           for i := 1 to min (m,r-k) do
                begin
                     y := f(y);
                     q := q |x-y| mod N
                end;
           G := GCD(q,N);
           k := k+m
      until (k ≥ r) or (G > 1);
      r := 2xr
until G > 1;  .
if G = N then
      repeat
           ys :=f(ys);
           G  := GCD(|x-ys|,N)
      until G > 1;
if G = N then {failure} else {success}.
```

In the section on the RSA PKC, a method of attack [proposed by Simmons and Norris (1977) and Herlestrom (1978)] which could lead to factoring was given with a method of selecting primes which makes the probability of success of such an attack small. This requires a proof that the numbers selected are primes. Two methods of doing this are described here.

The first method (Brillhart et al. 1975) of proving N a prime is based on a factorizing of N-1. If N is a prime, and $1<a<N-1$,

$$a^{N-1} \equiv 1 \pmod{N} \qquad *.$$

If there is an a that satisfies*, we say N is a pseudo prime to the base a, abbreviated N is a pspa. If there is an a, $1<a<N-1$, such that $a^{N-1} \not\equiv 1 \bmod N$, then N is composite. For most composite numbers this is true.

Step 1 - Factoring N-1

We now write $N-1=FR$, where F is completely factored, $F=\Pi P_i^{\alpha_i}$ with P_i distinct primes and g.c.d.$(F,R)=1$. It is often the case that a number of factors of N-1 can be found easily. If $(N/2)^{1/3}/F$ is small, say less than m, then test to see if $kF+1$ divides N-1, for $k=1,2,\ldots m-1$. If so, another factor is obtained and F is increased. If not, proceed to step 2. If F is not large, it is necessary to examine R for factors. If R can be proved prime, then N-1 is completely factored. If it proves difficult factoring N-1, use the second method.

Step 2

If N-1 is completely factored, or factored as just described, it is sufficient to find an a_i for each P_i which divides F, so that N is a $pspa_i$ and

$$\text{g.c.d.}(a_i^{(N-1)/p_{i-1}},N) = 1,$$

$$s = 0, \text{ or } r^2 - 8s \text{ is}$$

a square, where r and s are defined by $(N-1)/F=2Fs+r$, $1\le r<F$.

The a_i can be found by choosing a so that $(a|N)=-1$, where $(a|N)$ is the Jacobi symbol, and checking if

$$a^{N-1} \equiv 1 \pmod{N} \qquad (1^*)$$

$$\text{and } a^{(N-1)/p_i} \equiv b_i \equiv 1 \pmod{N}. \qquad (2^*)$$

If 2* is not satisfied for some p_i, find a new a. Such a's constitute a large percentage of the numbers $1<a<N-1$ whenever N is a prime.

The second method is Rabin's (1980) modification of the algorithms of Solovay and Strassen (1977) and Miller (1976). Although no proof exists[3] that this method proves that a number is prime, it may be adequate for this application. The algorithm follows.

1. Test N for small prime divisors. If none is found go to 2.
2. Choose b randomly from $1<b<N$.
3. Calculate $b^{n-1}(\text{mod } N)=C$. If $C\neq 1$, N is composite. If $C=1$, go to 4.
4. Calculate $N-1=2^k m$, where m is odd, and
   $X_o = b^m(\text{mod } N)$,
   $X_i \equiv X_{i-1}^2 \ (\text{mod } N)$, $1\leq i\leq k$.
   If, for some i, $X_{i-1}=N-1$, and $X_i=1$, then N is composite.
5. If N is not shown composite above, return to 2 and choose another b. Repeat until N is shown composite, or j times. After j repeats, the probability that choosing N as a prime when actually it is composite is less than $1/2^{2j}$. Note that this is not the probability that N is composite since it is either composite or not. Taking j=50 should usually suffice.

The effects of implementing the RSA-PKC with two numbers P_1 and P_2 thought to be primes, but actually composite, are illustrated by Blakely and Blakely (1979). In most cases, the error is discovered by encrypting and decrypting several messages.

The methods given in this appendix portray a difficult, but not impossible, task for the cryptanalyst.

APPENDIX A. REFERENCES

Blakley, B., and G.R. Blakley (1979b), Security of number theoretic public key cryptosystems against random attack, III, Cryptologia 3(2), pp. 105-118.

Brent, R.P. (1980), An improved Monte Carlo factorization algorithm BIT 20, pp. 176-184.

Brillhart, J., D.H. Lehmer, and J.L. Selfridge (1975), New primality criteria and factorization of $2^m\pm 1$, Math. Comp., 29, pp. 620-647.

Guy, R.K. (1975), How to factor a number, Congressus Numerantium XVI, Proceedings Fifth Manitoba Conference on Numerical Mathematics, Winnipeg, pp. 49-89.

[3]It has been reported (Science, 1980) that such a proof has been given, although I am not aware of its publication as of this date.

Herlestrom, Tore (1978), Critical remarks on some public-key cryptosystems, BIT, 18, pp. 493-496.

Lehman, S. (1974), Factoring large integers, Math. Comp., 28, pp. 637-646.

Lehmer, D.H. (1953), The sieve problem for all-purpose computers, Math. Comp., 7, pp. 6-14.

Lehmer, D.H. (1966), An announcement concerning the delay line sieve, DLS-127, Math. Comp., 20, pp. 645-646.

Miller, G.L. (1976), Riemann's hypothesis and tests for primality, Jour. Computer and System Science 13, pp. 300-317.

Miller, J.C.P. (1975), On factorization, with a suggested new approach, Math. Comp., 29, pp. 155-172.

Morrison, M.A., and J. Brillhart (1975), A method of factoring and the factorization of F_7, Math. Comp., 29, pp. 183-205.

Pollard, J.M. (1975), A Monte Carlo method for factorization, Nordisk Tidski Informationsbehandling (BIT), 15, pp. 331-334.

Rabin, M.O. (1980), Probabilistic algorithm for testing primality, Journal of Number Theory, 12, pp. 128-138.

Simmons, G.J., and M.J. Norris (1977), Preliminary comments on the MIT public key cryptosystems, Cryptologia, Vol. I, Oct., pp. 406-414.

Solovay, R. and V. Strassen (1977), A fast Monte Carlo test for primality, SIAM J. of Computing, 6, pp. 84-85.

APPENDIX B. EXAMPLES

The first example is a small example of the knapsack PKC. The original set is given by

$$a'_i = 2a_i + 3, \quad i = 1,2,\ldots,8;$$

$$a'_1 = 2.$$

Although this is a small set, it will illustrate the method. We form the new set

$$a_i = 1019a'_i \pmod{2111}$$

giving the first row in Table B-1. S is the sum of the odd numbered a_i. The rows below 8419 give the values

$$b_{ii} \equiv W_i a_i \equiv 1 \pmod{8419} \quad i = 1,2,\ldots,8$$

$$b_{ij} \equiv W_i a_i \pmod{8419} \quad j \neq i$$

$$s_i \equiv W_i S \pmod{8419}$$

following the method of attack suggested by Herelstrom (1978). The rows below 17389 give the same reduction using (Modulo 17389). It is seen that none satisfy

$$b_{ij} > \sum_{k \neq i} b_{kj}.$$

A total of 32 different moduli were tested with no success. Other, larger examples were also tested with an extremely small ratio of successes.

Some Examples of Implementation of the RSA-PKC

Table B-2 gives some examples of primes found using the methods described in Section 4. Note that the input primes in examples 4 and 5 and those in 6 and 7 are reversed, resulting in greatly different execution times.

Two 120-digit primes were found by using p and q and the reverse pair. The run times were 243.543 s and 1157.057 s, the longest time required for any run. The k required was approximately 122000 for this case. Thus, it appears that, if one pair does not give a prime in a reasonable time, the reverse pair should be used.

Table B-3 shows an example of encryption and decryption using the RSA-PKC with a 55-digit modulus. In this example, the message characters are converted to a 2-digit number and encrypted in blocks of 54 digits. The same message is encoded in each case, except for an added space at the beginning of the message for each repeat. Note the difference in the coded output blocks. The computer time required for this example was 1.529 seconds. Examples with a modulus of 210 digits were

Table B-1. An Example of a Knapsack PKC and One Possible Cryptanalysis Method

AI	A2	A3	A4	A5	A6	A7	A8	S
2038	800	435	1816	356	1658	40	1026	2869
8419								
1	5916	5953	2653	2801	4768	7031	4809	7367
4191	1	3526	5643	2905	4401	421	7852	2624
2482	776	1	1088	6407	5481	6774	6215	7245
7632	8234	4372	1	1391	2458	6305	3867	2862
526	5205	7829	6343	1	7525	2365	3834	2302
7100	1209	2920	2997	1422	1	3849	4855	6872
8049	20	3168	3413	7586	3830	1	5498	1966
5828	2643	7804	4400	8206	5204	1395	1	6395
17389								
1	8328	1920	820	9966	16912	7372	11724	1870
14305	1	8369	9914	8608	10392	9564	12304	6068
15315	12394	1	3442	6037	15514	12792	11635	16756
2778	7814	12726	1	2260	13847	12563	17064	12938
2448	7036	5130	7625	1	14756	14263	8502	4453
8895	420	2402	7909	15837	1	21	3147	9766
7876	20	10879	7001	15659	16561	1	2634	17026
12137	12068	1780	5832	16847	1188	7559	1	3545

Table B-2. Examples of Timing for Finding Primes for Use in the RSA-PKC

Input P Q	Time	Output P' 2P'+1
28001 46447	0.673 CP s	40 1125581401 80 2251162803
32609 48611	1.613 CP s	328 6366585511 657 2733171023
328 63665 85511 657 27331 71023	0.527 CP s	47088895 2578307107 2006468109 94177790 5156614214 4012936219
40 11255 81401 80 22511 62803	1.35 CP s	3095749 3247283877 1491251281 6191498 6494567754 2982502563
80 22511 62803 40 11255 81401	14.955 CP s	44492546 9477078230 9790469083 88985093 8954156461 9580938167
61 91498 64945 67754 29825 02563 30 95749 32472 83877 14912 51281	13.844 CP s	15310861 4171742946 0134117505 3359153702 1119121215 0316230689 30621722 8343485892 0268235010 6718307404 2238242430 0632461379
30 95749 32472 83877 14912 51281 61 91498 64945 67754 29825 02563	39.505 CP s	47339466 1093353440 2402658255 7268355969 5853426706 9186042569 94678932 2186706880 4805316511 4536711939 1706853413 8372085139
3095749 32472 83877 14912 51281 470 88895 25783 07107 20064 68109	26.351 CP s	196243056 4980088950 6001854158 5602538829 8437364072 4590910843 392486112 9960177901 2003708317 1205077659 6874728144 9181821687

Table B-3. An Illustration of One Implementation of the RSA-PKC

```
      BLKSIZE
   54.
      ENCODING EXPONENT
   163819.
      DECODING EXPONENT
   5924 7541203997 5114808016 8158247361 3334064474 9569121539.
      MODULUS
   14577 5415696635 2012326857 6183254506 6494747473 7831897629.

      INPUT TEXT
  EACH SUCEEDING MESSAGE WILL START WITH AN EXTRA SPACE.

 CODED BLOCKS
0341869831 4622250089 4123889061 1993100657 7039316148 47840
0319917704 5864601692 4481698538 7419748994 9337933862 88678

 DECODED TEXT
EACH SUCEEDING MESSAGE WILL START WITH AN EXTRA SPACE


      INPUT TEXT
   EACH SUCEEDING MESSAGE WILL START WITH AN EXTRA SPACE.

 CODED BLOCKS
0783913903 5985985343 8844379574 5230255343 1160651772 43501
0394977824 5790857971 9330064353 9487972442 2608759355 64592

 DECODED TEXT
 EACH SUCEEDING MESSAGE WILL START WITH AN EXTRA SPACE


      INPUT TEXT
    EACH SUCEEDING MESSAGE WILL START WITH AN EXTRA SPACE.

 CODED BLOCKS
0819819166 6210244319 0649191255 8791622974 7662566782 18925
0904535100 9514469651 6954870403 7524057453 2423031586 09977
0403096251 6564720351 6140109090 5341670035 5575349323 22997

 DECODED TEXT
  EACH SUCEEDING MESSAGE WILL START WITH AN EXTRA SPACE

      INPUT TEXT
     EACH SUCEEDING MESSAGE WILL START WITH AN EXTRA SPACE.

 CODED BLOCKS
0865904713 0411130643 7207784950 7199139875 0354315449 68896
0203655645 7888476912 6183150526 2165847252 6985926850 20809
0648393067 0371315329 4275471784 1465460103 3944600613 62429

 DECODED TEXT
   EACH SUCEEDING MESSAGE WILL START WITH AN EXTRA SPACE
```

run, with an average time of 0.83 s for encoding and decoding the 105-character message. This would correspond to a bit rate of approximately 759 b/s.

Some Examples of Factoring

We present examples of several of the factoring methods here, using the number $N = 267997 = 263 \cdot 1019 = PQ$.

1. Find $([\sqrt{kN}] + j)^2 \equiv x^2 \pmod{N}$ such that $X^2 < N$.

Table B-4 lists the k and j, with $j < 100$ for the first 12 occurrances.

Table B-4. An Example of Factoring Using $([\sqrt{kN}] + j)^2 \equiv x^2 \pmod{N}$

k	j	k	j
3	8	20	19
5	10	21	77
7	61	24	61
8	81	27	23
12	15	33	2
16	1	39	6

2. Using Pollard's (1975) Monte Carlo method, the "epact" is listed in Table B-5 for several functions and several starting values. Here, $X_m = f(X_{m-1}) \bmod p$ and the epact is that m such that $X_{2m} = X_m$.

Table B-5. Epacts for 263 and 1019

f(x)	Xo	263	1019
$X^2 - 1$	2	4	39
	7	11	31
$X^2 + 1$	2	25	15
	7	25	7

3. A partial example of the Morrison and Brillhart (1975) method is shown in Table B-6. The table consists of three parts, each containing A_{n-1} and Q_n needed for the relationship $A^2_{n-k} \equiv X_n \pmod{N}$ and the auxiliary numbers for the

Table B-6. The Continued Fraction Expansion $\sqrt{k \cdot 267997}$

	n -1	$g \pm P_n$ -	Q_n RN	s_n -	R_n g	A_{n-1} 0	Factored Q_n
	1	1034	708	1	326	517	$2^2 \cdot 3 \cdot 59$
	2	708	327	2	54	518	$3 \cdot 109$
	3	980	164	5	160	1553	$2^2 \cdot 41$
	4	874	857	1	17	8283	857
k = 1	5	1017	21	48	9	9836	$3 \cdot 7$
	6	1025	473	2	79	212414	$11 \cdot 43$
	7	955	161	5	150	166667	$7 \cdot 23$
	8	884	828	1	56	241758	$2^2 \cdot 3^2 \cdot 23$
	9	978	67	14	40	140428	67
	10	994	604	1	390	63774	$2^2 \cdot 151$
	1	1792	1175	1	617	896	$5^2 \cdot 47$
	2	1175	618	1	557	897	$2 \cdot 3 \cdot 103$
	3	1235	1115	1	120	1793	$5 \cdot 223$
	4	1672	181	9	43	2690	181
k = 3	5	1749	422	4	61	26003	$2 \cdot 211$
	6	1731	253	6	213	106702	$11 \cdot 23$
	7	1579	1334	1	245	130221	$2 \cdot 23 \cdot 29$
	8	1547	285	5	122	236923	$5 \cdot 57$
	9	1670	719	2	232	242848	719
	10	1560	505	3	45	186625	$5 \cdot 101$
	1	2070	763	2	544	1035	$7 \cdot 109$
	2	1526	1089	1	437	2071	$3^2 \cdot 11^2$
	3	1633	656	2	321	3106	$2^4 \cdot 41$
	4	1749	857	2	35	8283	857
k = 4	5	2035	84	24	19	19672	$2^2 \cdot 3 \cdot 7$
	6	2051	473	4	159	212414	$11 \cdot 43$
	7	1911	644	2	623	65337	$2^2 \cdot 7 \cdot 23$
	8	1447	1401	1	46	75091	$3 \cdot 467$
	9	2024	67	30	14	140428	67
	10		441			267976	$3^2 \cdot 7^2$

continued fraction expansion of kN, for k = 1,3,4. The numbers are computed from the relations

$$[\sqrt{kN}] = g$$

$$g + P_n = s_n Q_n + R_n$$

$$g + P_{n+1} = 2g - R_n$$

$$Q_{n+1} = Q_{n-1} + s_n(R_n - R_{n-1})$$

$$A_n = s_n A_{n-1} + A_{n-2} \pmod{N}$$

with the initial values

$$A_{-2} = 0,\ A_{-1} = 1,\ Q_{-1} = k_n,\ R_{-1} = g,\ P_0 = 0, \text{ and } Q_0 = 1.$$

For each k, values of A_n and Q_n were calculated up to n = 20. For k = 1 and 3, no set of factored Q's satisfied

$$1 < \text{g.c.d}\ (\pi A_{i-1} \pm \sqrt{\pi Q_i}, N) < n.$$

However, for k = 4, we find, for n = 2, $A_i - \sqrt{Q_2} = 2 \cdot 1019$ which factors N.

This is fortuitous, and hence does not require keeping the factors table which is usually required (see Morrison and Brillhart, (1975), for details). However, it does illustrate what a lucky choice of k can produce.

APPENDIX B. REFERENCES

Herlestrom, Tore (1978), Critical remarks on some public-key cryptosystems, BIT, 18, pp. 493-496.

Morrison, M.A., and J. Brillhart (1975), A method of factoring and the factorization of F_7, Math. Comp., 29, pp. 183-205.

Pollard, J.M. (1975), A Monte Carlo method for factorization, Nordisk Tidski Informationsbehandling (BIT), 15, pp. 331-334.

APPENDIX C. MULTIPLE PRECISION SUBROUTINES

The package of 31 subroutines described below was written in Compass and Fortran extended for use on the CDC 6600, CDC 7600, and CYBER 71/170 series computers. It was designed and written for speed and ease of use. All subroutines are Fortran callable.

The subroutines permit the user to work with integer variables and integer constants of arbitrary size (limited only by the the available memory). These multi-precision (MP) integers are represented by regular Fortran integer variables, thereby freeing the user from concerns such as storage space and internal structures. The only special requirement is for a reserved MP array (defined once in the main program) where all MP numbers can reside.

This section will identify all 31 subroutines, the category to which each belongs, and the basic operation or function performed by each.

INITIALIZATION

MPINIT — Initializes the dynamic storage parameters and the reserve MP array. Necessarily, it must be the first MP routine called.

MPSET — Initializes an MP number to a desired value in the range -(2**59 - 1) to 2**59 - 1.

INPUT/OUTPUT

MPIN — Reads an arbitrarily large integer from logical unit 5 and sets a specific MP number to this value.

MPOUT — Outputs a specific MP number of arbitrary length to logical unit 6, while allowing control over which columns are to contain the output digits. A 10-character identifier may also be printed.

STANDARD ARITHMETIC

MPADD — Places the sum of two MP numbers in a third MP number.

MPSUB — Places the difference of two MP numbers in a third MP number

MPMULT — Places the product of two MP numbers in a third MP number

MPDIV — Determines both the quotient and the remainder resulting from the division of two MP numbers.

MODULAR ARITHMETIC

For all of the routines in this category, it is necessary to provide a non-zero MP number N as modulus.

MPMODAD — Places the sum (modulo N) of two MP numbers in a third MP number.

MPMODSB — Places the difference (modulo N) of two MP numbers in a third MP number.

MPMODML — Places the product (modulo N) of two MP numbers in a third MP number.

MPMODDV Determines whether the quotient (modulo N) of two MP numbers exists, and if it does, places the quotient in a third MP number

MPMODPW Determines the result of raising (modulo N) one MP number to a power represented by another MP number. The result is placed in a third MP number.

NUMBER MANIPULATION

MPMOVE Sets one MP number equal to another MP number.

MPSWAP Interchanges the values of two MP numbers.

MPSHFTL Multiplies one MP number by a given integer power of 2 and places the result in a second MP number (Thus performing a left shift for positive powers).

MPSHFTR Divides one MP number by a given integer power of 2 and places the quotient in a second MP number (thus performing a right shift for positive powers).

MPCMP Compares the values of two MP numbers and returns the result (<, =, or >) in a regular (non-MP) integer variable.

MPLEN Returns (in two regular integer variables) the current length in words and the physical length in words of a given MP number.

MPGETWD Returns in a regular integer variable the value of a specific word of a given MP number.

SIGN MANIPULATION

MPSIGN Returns in a regular integer variable the sign (either -1, 0, or 1) of a given MP number.

MPABS Changes (if necessary) the sign of a given MP number to yield a positive number (thus an absolute value routine).

MPCHGS Changes the sign of a given MP number.

SPECIAL FUNCTIONS

MPSQRT Places the square root of a given MP number in a second MP number.

MPGCD Places the greatest common divisor (g.c.d.) of two MP numbers in a third MP number.

MPGCDXY Places the g.c.d. of two MP numbers (say A and B) into a third MP number (say C). In addition, it returns two MP numbers X and Y with the property that A*X + B*Y = C.

STORAGE AND ERROR MANIPULATION

There should be little need for the user to call the routines listed under this category, since they are all called automatically when needed.

MPALLOC	Establishes a contiguous block in the reserved MP array for storage of a given MP number.
MPFREE	Releases the assignment of a block in the reserved MP array.
MPPACK	Repositions within the reserved MP array all currently assigned blocks so that they are contiguous.
MPSQEEZ	Reduces the space allocated to each MP number so that its physical length is only 1 more than its current length. After this it repacks the reserved MP array.
MPERROR	Outputs error messages to logical unit 6 and then either returns control to the calling subroutine or halts execution of the main program.

FORM **NTIA-29**
(4-80)

U.S. DEPARTMENT OF COMMERCE
NAT'L. TELECOMMUNICATIONS AND INFORMATION ADMINISTRATION

BIBLIOGRAPHIC DATA SHEET

	1. PUBLICATION NO.	2. Gov't Accession No.	3. Recipient's Accession No.
	NTIA Report 81-79		

4. TITLE AND SUBTITLE

A CRITIQUE OF SOME PUBLIC-KEY CRYPTOSYSTEMS

5. Publication Date
August 1981

6. Performing Organization Code

7. AUTHOR(S)
William J. Hartman

9. Project/Task/Work Unit No.

8. PERFORMING ORGANIZATION NAME AND ADDRESS

U.S. Department of Commerce
National Telecommunications and Information Admin.
Institute for Telecommunication Sciences
325 Broadway, Boulder, Colorado 8030

10. Contract/Grant No.

11. Sponsoring Organization Name and Address

NTIA/ITS
325 Broadway
Boulder, CO 80303

12. Type of Report and Period Covered

13.

14. SUPPLEMENTARY NOTES

15. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.)*

Several of the public-key cryptosystems that received considerable publicity are examined. The uses, implementation, and potential crypto-analytic attacks are discussed. Since most of the suggested systems depend on the manipulation of large numbers, special multiprecision computer programs have been developed to demonstrate system implementation and crypt-analytic attacks. Examples of the use and timing of these programs are included.

16. Key Words *(Alphabetical order, separated by semicolons)*

Public-key; number theory; cryptology; algorithms

17. AVAILABILITY STATEMENT

☒ UNLIMITED.

☐ FOR OFFICIAL DISTRIBUTION.

18. Security Class. *(This report)*
Unclassified

20. Number of pages
45

19. Security Class. *(This page)*
Unclassified

21. Price:

☆ U.S. Government Printing Office: 1980—678-495/529