Towards a Practical Public-key Cryptosystem

by

Loren M Kohnfelder

Submitted in Partial Fulfillment

of the Requirements for the

Degree of Bachelor of Science

at the

Massachusetts Institute of Technology

May, 1978

Signature of Author Loren M Kohnfelder
Department of Electrical Engineering, May 10, 1978

Certified by Leonard Adleman
Thesis Supervisor

Accepted by David Cole
Chairman, Departmental Committee on Theses

ARCHIVES
MASS. INST. TECH.
MAY 25 1978
LIBRARIES

Table of Contents

List of Figures

List of Algorithms

Towards a Practical Public-key Cryptosystem

by

Loren M Kohnfelder

Abstract.

One of the central requirements for traditional cryptographic communication, the secure transmission of a key between communicants, has recently been shown unnecessary. The resulting cryptosystems in which keys are transmitted via public communication channels are called *public-key* cryptosystems. This paper briefly describes the methods that have been presented for achieving public-key communication but is mainly concerned with the method recently published by Rivest, Shamir, and Adleman (RSA).

The trap door permutation which underlies the RSA algorithm is presented and its computability and difficulty to invert discussed. The method itself is derived together with parameter construction techniques to insure security. The problems of signatures and key representation are discussed.

The concept of *certificates* is presented to reduce the active role of a public authority administering the public keys. A specification is given for a possible implementation of the method in a large communication system. Evaluation of the method indicates that it is presently of practical interest, although specialized hardware is probably necessary to achieve high speed transmission rates. Convincing evidence for the security of the system exists although further research is in order to increase confidence in the method.

Thesis Advisor: Len Adleman, Assistant Professor.

Introduction. This thesis is concerned with the problem of actually achieving a practical communication system using public-key cryptography. The public-key paradigm seems suited for communications applications requiring security. The differences between traditional and public-key techniques are often quite subtle; this paper endeavors to shed some light on this issue. Also a summary and comparison of published public-key cryptographic methods seemed lacking in the literature and is presented herein.

An attempt is made to present the RSA method in sufficient detail so as to allow its successful implementation. Where the original paper is conceptually complete, this paper presents a "cookbook" approach, while at the same time clearly explaining the motivation behind the various peculiar details of the method. The ideas on signature reblocking, key representation, and message alteration are the author's.

The concept of certificates was developed by the author as an aid in simplifying the communication problems encountered when implementing the method. The suggested RSA implementation is intended to give an example of the many details to be considered in system design. The proposed implementation is a viable one and as confidence in the method grows, a system such as the one described could be put into use.

I. Overview of Public-key Cryptosystems.

A. Motivation for Public-key Cryptosystems.

Traditional methods. Information has long been regarded as a valuable resource, and its dissemination sometimes must be controlled. When people communicate over any indirect medium (by courier, telephone, etc.) they may wish to send messages which can only be understood by the authorized individuals. The goal of any cryptosystem is to provide such a secure method for communication.

Cryptosystems are based on general algorithms which are applied to particular randomly chosen parameters known as keys. The so-called traditional methods are those whose security is jeopardized by the knowledge of the keys in use by anyone other than the authorized communicants. Public-key methods remain secure even when an enemy learns the key. Of course there must be some necessarily secret information in a public-key cryptosystem but it must be known by only one communicant.

Any traditionally encrypted communication must be prefaced by the secure transmission of a key. The key must be sent from one communicant to the other without the enemy either knowing the key or sending spurious keys via the same channel. Then to send a message, one communicant uses the key and a commonly known algorithm to encrypt the message. The encrypted message is then transmitted over an insecure channel. Should the enemy learn the contents of this transmission it would be meaningless without also knowing the key. When the authorized communicant receives the transmission, using the key and the inverse algorithm to that used by the sender, the original message is easily recovered.

Weaknesses of traditional methods. The very quality that distinguishes traditional from public-key methods, the need to maintain keys secretly, is a major weak point of traditional systems. When secure communication over insecure channels must take place, the task of safely transmitting a key is not an easy one. Keys are sent through the necessarily high-security channels when reasons of time or economy prohibit sending the messages by these channels.

But getting a key to its intended user is only part of the problem. The key must be constantly kept secure lest the security of all transmissions, future and past, be violated. Numerous practical problems arise in maintaining key security. Keys no longer in use must be destroyed by an individual with sufficient authority to do so reliably. The whereabouts of all copies of keys in use must always be known by some high authority. If the security of a key is jeopardized there may be great expense and delay in contructing a new key and securely sending it to the authorized communicants.

There is another problem that arises from the private key restriction of traditional methods. Consider a large group of communicants within which pairwise secure communication must occur (such as the members of a large electronic mail system). Since the transmission between any pair of individuals is secure from all others in the system, each pair of individuals must use a unique key. If the group consists of N communicants, there must be N-1 distinct keys known by each communicant (one for every other communicant in the system). Then there must have been the total transmission of $(N^2-N)/2$ keys through secure channels from every communicant to every other. Since public-key cryptosystem keys need not be secret, one key will serve for all communication with a certain communicant. Hence in a large group within which pairwise security is needed, only N keys need exist and furthermore need not be transmitted via secure channels.

Traditional methods redeemed. Despite the drawbacks to traditional methods, in many real world situations they are quite adequate. Since traditional methods have been used for so long compared to the recent arrival of public-key methods just a few years ago, they are much better understood and much experience about them has been gained. So far no proofs of the unbreakability of any public-key cryptosystem have been given. In this sense traditional methods are more reliable since there exist much more exact estimates of their difficulty to break.

If any code is impossible to break it is a traditional one, the one-time pad system. Using this system each word or phrase is translated into some random expression found in a large

codebook. However, each new instance of a message word is encoded into a new code word. Since a code word once used is never repeated, there is no pattern to the encoded message. When it is possible to send securely such a large codebook and rely on its subsequent security, this is a very desirable method.

Good traditional encryption techniques exist which are highly efficient in both data compactness and speed. Also, traditional methods are often quite simple computationally and therefore more convenient to implement. When the need for security is only moderate, such a traditional approach is useful.

It has been suggested [1] that a public-key method be used to bootstrap into a traditional encryption method. In such a system a key is randomly generated for each conversation by one of the communicants. Using public-key methods, this key is securely sent to the other communicant over an insecure channel otherwise not suited for the transmission of a traditional key. That key is used from there on with a traditional method to continue secure communication in a faster and more efficient way.

B. Conceptual Basis of Public-key Cryptosystems.

The public-key criterion. The great weakness of traditional encryption methods is that they require the communication of a key between communicants over a secure channel. Merkle [5] shows this requirement of secure key transmission to be an element of the basic paradigm upon which cryptography has long been based. Public-key cryptography is therefore achieved by removing this requirement from the paradigm. Thus there is a single condition on the key channel for public-key cryptosystems.

> The enemy may not transmit messages over the key channel without detection.
>
> Furthermore any modification of messages transmitted over the key channel is also detectable by the authorized communicants.

Implicit in this condition is that the enemy can have complete knowledge of the transmissions over the key channel and that such eavesdropping may be undetectable by the authorized users of the channel. This eavesdropping ability of the enemy's was explicitly forbidden in the traditional paradigm, and the deletion of this clause is the fundamental distinction between traditional and public-key cryptography.

How public-key communication can work. A general schema for public-key cryptographic algorithms has been developed by Diffie and Hellman [1] based on their concept of trap door one-way functions. The traditional notion of a key and an algorithm is conceptually replaced by a function. To clarify the concept and to demonstrate how public-key methods differ from traditional, consider the traditional encryption scheme.

A given pair of individuals wish to communicate securely. They have secretly communicated a key between them. This key allows each of them to compute effectively two functions, say E and D, the encryption and decryption functions. E maps messages into code space; D maps from code space into messages. Since D decodes messages encrypted with E it is required that

$$(\forall M \text{ messages})[D(E(M)) = M]$$

The key must be kept secret from the enemy since both D and E can be computed from it.

Diffie and Hellman suggest that two keys be used, a different one for sending messages to each communicant. Now it may be possible to construct keys that only allow E to be computed easily from them. Such a key is generated by the communicant who already knows the decoding function. The decoding function need never be given to the other communicant; only a way of computing the encryption function needs to be sent. Therefore when the enemy knows the key that only gives him the ability to send messages! Returning from the concept of keys to functions, the basic scheme is described.

Trap door one-way functions. The concept of revealing the encryption function without giving away any practical method of computing the decryption function is essential to public-key methods and is a condition to the definition of a trap door one-way function. Formally these functions are characterized by three conditions. F is a trap door one-way function with inverse F^{-1} if and only if:

(i) Both F and F^{-1} are easy to compute.

(ii) $(\forall M \text{ messages})[F^{-1}(F(M)) = M]$.

(iii) Revealing how to compute F does not also reveal any reasonable way of computing F^{-1}.

Condition (ii) gives the property necessary for the use of F as an encryption function, i.e. that coded messages can in fact be decoded. Condition (i) is necessary to insure that it is practical to do the encoding and decoding computations. Condition (iii) also relies on (i) to give the correct sense to the trap door properties these functions exhibit. It would be possible to satisfy (iii) alone with a function whose inverse is not effectively computable. That is, if there exists no reasonable way of computing the inverse, then certainly revealing how to compute the function reveals no such way. With (i) also satisfied it is implied that the difficulty in computing F^{-1} with only knowledge of F is due to the difficulty in determining an algorithm for F^{-1}, not in actually performing the computation.

Further restrictions must be placed on trap door one-way functions for them to be useful in

the construction of a public-key cryptosystem. The size of the range of F cannot be immensely larger than the size of its domain. Roughly speaking, the ratio of the size of the range to that of the domain is the data expansion coefficient of the encryption function. Practical cryptosystems require that this coefficient be rather small, certainly less than one hundred to be at all useful. Ideally this coefficient is near one.

Not only must the range not be too big, but it must not be too small either. A small range implies a small domain. When the domain is small the inverse of the function becomes computable by trial and error. A domain of greater than 10^{20} elements is always a good measure, and should be over 10^{30} for high security applications.

A very important special case of the trap door one-way function is the trap door one-way permutation. This corresponds to the condition that the image of the function is exactly its domain. Formally a fourth condition can be added to those for trap door one-way functions to specify the permutation case.

(iv) $(\forall M \text{ messages})[F(F^{-1}(M)) = M]$.

Attacks on public-key cryptosystems. Since public-key communication systems allow the enemy to have knowledge of the encryption algorithms as well as all ciphertext, forms of attack on the system are possible that do not apply to traditional method systems. It has been defined to be infeasible to compute the inverse function easily from the forward function algorithm, therefore that attack is not considered here. There exists inherently for all public-key cryptosystems a weak form of attack. This is the trial and error method mentioned above, and is why the domain of the encryption function must be large. Since in any public-key cryptosystem the enemy may have knowledge of the encryption function, the following algorithm will always break the code.

Algorithm 1. To determine $F^{-1}(C)$ given F (the function) and C (the code word):

```
1.  M ← 0                          {Initialize}
2.  if F(M) = C then goto 4        {Test solution}
3.  M ← M+1 ; goto 2               {Try a new solution}
4.  output M                       {Output answer}
```

Fortunately this attack is avoidable even with unlimited gains in computer technology. This is because increasing the size of the domain of the encryption function by a few orders of magnitude only slightly increases the time to compute the function while greatly increasing the time to run the trial and error decryption function. By increasing the domains so that they safely exceed the computational limitations of the day, the trial and error attack is always avoidable.

Given ideal conditions, no other attack is possible. However there are certainly potentially exploitable weakness in real public-key cryptosystems. Probably in any high security public-key cryptosystem the easiest approach is to steal the decryption function. Also the trial and error algorithm can be modified to jump randomly around within the domain. While this is not likely to succeed, there does always exist some probability of finding a particular instance of the decryption function. In cases of extremely sensitive information the small chance of breaking the code might be worth taking if the rewards for obtaining the information are great enough.

Another form of attack is possible unless care is taken in the choice of messages sent. Messages must always be of arbitrary contents; the enemy should never be able to guess what possible messages might be. Suppose the enemy thinks a pair of communicants may be choosing days on which a particular event is to occur. If simple messages are used in the conversation (such as "5/6/78", etc.) then the enemy can encrypt a large set of such dates and decypher any simple date response by matching the transmitted code to the set of encrypted dates. A field of perhaps ten random characters should be appended to such simple messages

to counter this attack.

Messages should also contain either information about date and time of transmission or some sequence numbers. Sequencing is good because lost messages are easily detected. Without either of these techniques to make messages unique it is possible for the enemy to record messages and retransmit them subsequently. Such retransmissions are indistinguishable from legitimate transmissions.

C. Public-key Algorithms.

This section briefly describes the methods of implementing public-key cryptosystems published or known to the author as of early 1978. The intent here is to show the character and strengths of the methods, not the details. The RSA algorithm is extensively treated in Section II. The references cited after the section titles give the details of the methods.

Merkle. [5] This method, first developed in 1975, was primarily important in demonstrating that it was possible to achieve public-key encryption. The paper in which the method is described elegantly states some of the basic principles involved in public-key communication: conditions on the key channel, and the necessary introduction of random information the selection of a key.

The fundamental notion of the method is the *puzzle*. A puzzle is an encrypted message which can be decrypted (without knowing the key) by trial and error in some reasonable amount of time. These puzzles are randomly generated with the key space restricted so that the puzzle is breakable, but not too easily. The idea is that each communicant will have to solve about one puzzle (or do some roughly equivalent amount of work), while an enemy would have to solve many puzzles to break the code.

Consider communicants A and B establishing communication impervious to the efforts of the enemy E. Communicant A generates N puzzles each designed to be of difficulty O(N). Each puzzle, once solved, contains a key to be used for traditional cryptographic communication thereafter. Also each solved puzzle contains an identifier unique among the puzzles and randomly generated. Supposedly the creation of all of these puzzles requires O(N) effort.

These puzzles are transmitted to B (and possibly to E) who randomly chooses one of them, ignoring all the rest. Spending O(N) effort, B solves the selected puzzle, extracting the contained key and identifier. The identifier is then transmitted back to A who immediately matches it up through a table made during the creation of the puzzles which maps an identifier into the key in the corresponding puzzle. Thus, both A and B have spent O(N) effort and

arrived at a mutually known key.

The only hope for E to break the code is to break puzzles randomly until the identifier sent to A is found. On the average, E will have to break (N+1)/2 puzzles in order to do so, thus the expected amount of effort facing E is $O(N^2)$ since each puzzle requires O(N) effort. Thus if N=2000, E is faced with an expected amount of work approximately a thousand times that expended by either of the communicants.

Clearly this method is not practical when compared to known $O(2^N)$ methods, yet it is of limited use. If one is willing to expend a day of computation on a large scale computer in puzzle generation and then another day at a distant site solving one of those puzzles, the enemy is faced with several years of puzzle solving before standing a better than even chance of breaking the code.

Most importantly, the method does allow public-key cryptography and was the first actual example of how information can be effectively hidden when transmitted over a public channel.

Merkle-Hellman. [6] This method is based on the knapsack problem which is used as the trap door one-way function. The knapsack problem is NP-complete and in general believed to require exponential effort to solve. As is typical of public-key cryptosystems, a trial and error attack is possible; such an attack on the average requires $O(2^N)$ effort. The basic principle behind the method is that knapsacks are constructable such that with secret information they are quickly solvable. Of course this secret information must not be derivable from the knapsack itself.

The knapsack problem is one of selecting a subset from a given set. A given problem consists of a vector of numbers and a single number. The solution to the problem is a bit vector with a bit on in each position corresponding to a number in the vector that is used in forming a sum of the given number. More formally, if the problem is $\langle\{x[i]\}, S\rangle$ then $\{a[i]\}$ is a solution if

$$S = a \cdot x = \Sigma\, a[i]\, x[i]$$

where "$\cdot$" denotes the dot product. It is well known that knapsacks exist which are easily

solved. For instance a knapsack which has the property that each element is greater than the sum of all elements less than it can be solved quickly.

The creator of the problem randomly chooses such an easily solved knapsack. Then two large relatively prime numbers are chosen such that the larger is greater than the sum of the elements of the knapsack. Using the other number and its inverse modulo the larger number, the easy to solve knapsack is transformed into an apparently difficult one. Only when the secretly chosen parameters are used to convert a given problem back into the easily solved system is a solution effectively found. Clearly anyone knowing the knapsack can make problems by simply adding up elements of the vector. Thus only through the special knowledge of the creator of the problem is the inverse of the encryption function computable effectively.

Merkle and Hellman go on to describe multiplicative knapsacks as well as issues of signatures and compressing the knapsacks. The additive knapsack problems cost a two for one data expansion. Unfortunately, to achieve secure communication requires quite a big knapsack (Merkle and Hellman say 2×10^4 bits) although the actual secret information is only a few hundred bits. There is of course the threat of new advances in the solution of knapsacks since no good lower-bound proofs are known. Such an advance seems unlikely at the present time, and this same problem of future advances plagues all known trap door one-way functions.

Rivest, Shamir, and Adleman. [11] This method is based on a trap door one-way permutation found in number theory. The fundamental theorem is due to Euler and Fermat and is described later in part II.A and in both [2] and [8].

Encryption consists of raising the message number to a given exponent modulo some large integer. Decryption consists of additionally raising the encrypted message to another (secret) exponent. The resulting composite function is the identity function, and thus the original message is revealed. The secret exponent is computable from the prime factorization of the modulus which is constructed as the product of random primes. Thus the constructor of the code knows the factorization, yet this factorization is kept secret since only the product is publicly revealed.

This method uses a trap door one-way permutation and thus has a data expansion ratio of one. Each communicant has a unique set of parameters to the basic algorithm denoted

<n,e>

the integer modulus and exponent for encryption. With each of these integers about 500 bits long the best known algorithms would take hundreds of years to break the code. Thus about a thousand bits of information are needed to represent the encryption parameters. Selection of these parameters is a somewhat complicated and time consuming process, but need only be done once in the "lifetime" of a communicant. Again there lurks the danger of future advances in the theory of computing prime factorizations which, if successful enough, could invalidate the technique.

D. Weaknesses in Public-key Cryptosystems.

Although the use of public-key methods alleviates many of the difficulties involved in key transmission and security, other problems still persist. Furthermore the public-key solution introduces new weaknesses not typical of traditional cryptosystems.

Remaining problems. The public-key approach is far from ideal. Although many of the remaining problems are not within the scope of cryptography, it must be realized that public-key cryptosystems are breakable. Any cryptosystem must be based on some secret information known only to authorized individuals. Should any unauthorized individual learn such secret information the system security is necessarily compromised. The actual task of distributing and maintaining such secret information is a formidable one and most certainly a potential weak point.

Although the enemy may eavesdrop on the key transmission channel, the key must be sent via a channel in such a way that the originator of the transmission is reliably known. For a public-key conversation to occur, each communicant must reliably learn the others' encryption function. They cannot simply ask each other because an enemy could impersonate one of the communicants and give a false encryption function whose inverse would be known to the enemy. Without trust in a third party and without a reliable-source line, keys cannot be learned and therefore public-key communication cannot be initiated. Also if a communicant loses his decryption function it cannot be cancelled and replaced by a new function without such a reliable medium.

The Public File. Public-key communication works best when the encryption functions can reliably be shared among the communicants (by direct contact if possible). Yet when such a reliable exchange of functions is impossible the next best thing is to trust a third party. Diffie and Hellman introduce a central authority known as the Public File. The Public File is a highly authorized entity that serves as a dynamic directory for all of the encryption functions in a given system. Each communicant registers once (in person) with the Public File before any

communication takes place. The Public File is given the name and corresponding encryption function of each communicant. In return, each communicant receives the encryption function of the Public File.

To learn the encryption function for a certain communicant, a signed message (see part II.C) is sent to the Public File containing the request. The Public File returns, signed, the requested information. The signature on the information request allows the Public File to record who is given what information. The signed response insures that the Public File is not being impersonated by an enemy.

The Public File solves many problems, but it is also a great potential threat to system security. An enemy that had broken the Public File encryption function could authoritatively pass out bogus encryption functions and thereby impersonate any communicant in the system. Even without breaking the Public File encryption function such impersonation is possible by tampering with the records kept by the Public File. The Public File could not work in high security applications since it would not be trusted. Consider a Public File coordinating all diplomatic communications in the world; who could reliably operate such an authority?

II. The RSA Method.

A. The Trap door function.

Preliminary number theory. The fundamental identity underlying the RSA method is a direct result of a deep theorem of number theory. Before the theorem can be presented, a few basic definitions and notational remarks are necessary.

Throughout this section all numbers and variables should be considered integers unless otherwise specified. Z(n) denotes the group of integers modulo n, or $\{ i : 0 \leq i < n \}$. The "greatest common divisor" (*gcd*) function denotes the largest integer dividing a set of integers. The gcd function is denoted by a set of numbers enclosed in parenthesis. Thus the gcd of "i" and "j" is written as

$$(i,j)$$

Two integers are ***relatively prime*** if their gcd is one. An important property of the gcd is that

$$(\forall m)(\forall n)(\exists k_m)(\exists k_n)[\ (m,n) = m\ k_m + n\ k_n\ ]$$

The Euler ***totient*** function, written $\phi(n)$, denotes the number of positive integers less than n which are relatively prime to n. The following rules define the totient function. The variable "p" denotes a prime.

$$\phi(p) = p-1$$

$$\phi(m\ p) = \phi(m)\ \phi(p) \qquad (m,p) = 1$$

$$\phi(m\ p) = \phi(m)\ p \qquad (m,p) = p$$

The Euler-Fermat Theorem. This fundamental theorem serves as the basis for the construction of the trap door one-way functions used in this method. It may be stated as

$$(\forall m)(\forall n)[(m,n) = 1 \ \Rightarrow\ m^{\phi(n)} \equiv 1 \pmod{n}]$$

using the notation described in the preceding paragraphs. From this theorem a slightly different form will be derived that, in a chosen special case eliminates the relative primeness condition of the original theorem.

Consider the case of n being the product of exactly two distinct primes.

$$n = p\,q \quad (p \neq q)$$

To prove the identity for all m (not just those relatively prime to n), it must be proven for (m,n) $\neq$ 1. Since the identity is to hold in Z(n), only values of $0 < m < n$ are considered. Since n is the product of two primes, if (m,n) is not one it must be one of those primes that factor n. Consider the case (m,n) = p. It follows from $m < n$ and (m,n) = p that (m,q) = 1. Therefore by Euler-Fermat

$$m^{\phi(q)} \equiv 1 \pmod{q}$$

Since $\phi(q)$ divides $\phi(n)$ by elementary properties of the totient, it follows that

$$m^{\phi(n)} \equiv 1 \pmod{q}$$

and similarly for all m it can be shown that

$$m^{\phi(n)} \equiv 1 \pmod{p}$$

Since for all $0 < m < n$ the identity holds both (mod p) and (mod q) it must be that

$$(0 < m < n) \Rightarrow m^{\phi(n)} \equiv 1 \pmod{n}$$

Multiplying both sides of the equation by m gives the desired form of the equation.

$$m^{\phi(n)+1} \equiv m \pmod{n}$$

Note that the above equation holds for m = 0 and therefore for all m in Z(n).

Derivation of the function. The final result of the previous paragraphs provides an identity function applicable to cryptography. Here it is shown how to construct an encryption and decryption function pair based on this result. The final part of this section is devoted to a discussion of the apparent trap door properties these constructed functions have.

To construct an encryption function it is first necessary to choose two large primes at random. The product of these primes shall be called n and limits the size of the message space. Using some conventional scheme, messages are mapped into Z(n). Large messages must be broken into sufficiently small segments.

From the known prime factorization of n, $\phi(n)$ is computed.

$$n = p\,q \qquad \phi(n) = (p-1)(q-1)$$

A random number relatively prime to $\phi(n)$ is chosen and designated d. This is the secret

information that will be necessary to decrypt messages sent with this code. Since $(\phi(n),d) = 1$, d^{-1} must exist uniquely in Z(n). It is computed and designated e.

The encryption algorithm is characterized by the public information $\langle n,e \rangle$. A message, M (in Z(n)), is encrypted by computing

$$M^e \pmod{n}$$

To decrypt this code word, C, the originator of the function need only compute

$$C^d \pmod{n}$$

To show that the result of the above computation is indeed M, it is necessary to recall the basic relationships among n, e, and d.

$$(n,e) = 1 \qquad e\,d \equiv 1 \pmod{\phi(n)}$$

Considering the decryption computation and substituting for C yields,

$$C^d \equiv (M^e)^d \equiv M^{e\,d} \pmod{n}$$

Since e is d^{-1} in $Z(\phi(n))$ it follows that

$$(\exists k)[\, e\,d = k\,\phi(n) + 1 \,]$$

Thus for some k, substituting in for the expression for the decoded code word gives,

$$C^d \equiv M^{k\,\phi(n)+1} \equiv M\,(M^{\phi(n)})^k \pmod{n}$$

But it has been shown previously that $M^{\phi(n)} \pmod{n}$ is one. Simplifying the above equation yields,

$$C^d \equiv M\,(1)^k \equiv M \pmod{n}$$

Although the appeal to Euler-Fermat was based on the assumption of m positive, the final result is clearly true when M = 0 as well.

The following table summarizes the conditions on the parameters and shows the formulas of the encryption and decryption functions.

Figure 1. Summary of the RSA Algorithm.

$n = p\,q$	(p and q both prime, $p \neq q$)
$(\phi(n),d) = 1$	$(1 < d < \phi(n))$
$e\,d \equiv 1 \pmod{\phi(n)}$	$(1 < e < \phi(n))$

To encrypt:	To decrypt:
$C = M^e \pmod n$	$M = C^d \pmod n$

Difficulty of inverting. To invert a given encryption function is to break the code. The inverse function is unique because $e^{-1} \pmod{\phi(n)}$ is unique in $Z(\phi(n))$. Thus determining the secret parameter d is exactly what is required to break the code.

One possible theoretical attack is to factor n. Knowing the factorization of n it is easy to compute $\phi(n)$ which allows direct calculation of d.

$$d = e^{-1} \pmod{\phi(n)}$$

Currently, no reasonable algorithm is known that quickly factors very large integers (say, over a few hundred decimal digits). Therefore, at the present time this attack is unfruitful. Furthermore it will be argued that any other attack on the system is computationally as difficult as factoring n. Thus when large values of n are used, until factoring techniques are vastly improved, the code will stand impervious to analytical attack.

Another method would be to somehow directly determine $\phi(n)$ without ever knowing the factorization of n. Should this be accomplished, again it would be quite easy to determine the secret parameter d. Computing $\phi(n)$, however, is as hard as factoring n. Given $\phi(n)$, the factors of n (p and q) are easily determined. Since

$$\phi(n) = (p-1)(q-1) = n + 1 - (p + q)$$

It is possible to determine $s = p + q$ from $\phi(n)$ as

$$s = n + 1 - \phi(n) = n + 1 - (n + 1 - (p + q)) = p + q$$

From this follows the difference squared, $r = (p - q)^2$

$$r = s^2 - 4n = (p + q)^2 - 4n = p^2 + 2n + q^2 - 4n$$

$$r = p^2 - 2n + q^2 = (p - q)^2$$

Now p and q are now computed directly from r and s to be

$$s + \sqrt{r} \qquad s - \sqrt{r}$$

Finally, perhaps some algorithm is discovered which computes the secret information, d, without ever learning $\phi(n)$ or the factorization of n. Knowing d it is possible to compute some multiple of $\phi(n)$ since

$$e\,d \equiv 1 \pmod{\phi(n)} \Rightarrow (\exists k)[\, e\,d - 1 = k\,\phi(n)\,]$$

Assuming Extended Riemann Hypothesis (ERH), Miller [7] has demonstrated that determining any multiple of $\phi(n)$ is polynomial time equivalent to factoring n.

$$\text{"factorization of n"} \approx_p \text{"computing } k\,\phi(n)\text{"}$$

That is, determining d from only n and e is approximately as hard as computing the factorization of n, and that if d can be determined in a reasonable amount of computation then n can be factored in a reasonable amount of time as well. Miller gives a probabilistic algorithm for computing $\phi(n)$ from any multiple of $\phi(n)$ which, assuming ERH, runs in a reasonable amount of time.

Thus it has been shown that any possible attack on the code is computationally as difficult as factoring n and would lead to an efficient factoring algorithm if a good attack strategy were discovered. While presently no useful lower bound is known for the factoring problem, it has been the subject of great effort by many mathematicians and so far no algorithms to factor large integers quickly have been found.

The NBS Data Encryption Standard was certified by the unsucessful expenditure of many man-years trying to break the code. Similarly, since the RSA method security rests on the computational difficulty of the factoring problem, the great effort to find fast factoring methods certifies it. Thus although the RSA method is not provably secure it is well certified according to current standards.

B. The Method.

The encryption process. For both encryption and decryption, the fundamental computational operation is exponentiation modulo a large integer. Consider a specific instance of the RSA cryptosystem with public information <n,e> and secret decoding information, d. Encryption of a message word, M, consists of computing

$$M^e \pmod n$$

Similarly, decrypting this code word, C, means computing

$$C^d \pmod n$$

Clearly efficient ways of exponentiating modulo n are required for effective use of the method.

Exponentiation in Z(n) is accomplished by repeated multiplication in Z(n). If the exponentiation were done over the integers and then reduced modulo n, the intermediate values would be of staggering magnitudes. Thus for reasons of storage and speed it is desirable to reduce each multiplication over Z(n).

Knuth [4] gives a well-known algorithm for exponentiating in a number of multiplications proportional to the logarithm of the exponent. The principle underlying the algorithm is that any a^m can be expressed as the product of terms of the form

$$a^{(2^i)} \quad (0 \le i \le \lceil \log_2 m \rceil)$$

The largest term in this set is computable in O(log m) multiplications with each smaller term being an intermediate value of the computation. The algorithm may be stated as follows.

Algorithm 2. To compute a^m.

```
1.  Z ← 1 ; T ← a ; E ← m          {Initialize}
2.  if odd(E) then Z ← Z T         {Multiply in a term}
3.  E ← ⌊ E + 2 ⌋                  {Halve exponent}
4.  T ← T T                        {Square term}
5.  if E > 0 then goto 2           {Loop until E = 0}
6.  output Z                       {Output answer}
```

The RSA method uses exponentiation in Z(n), so all multiplications shown above would be performed in Z(n). Clearly choosing the encryption exponent (e) to have only a few 1 bits in its binary representation will make this algorithm run significantly faster than if the exponent has a random binary representation.

Adleman points out that intermediate computations need not be reduced over Z(n), but rather over Z(kn) for any positive k. The final answer is computed by reducing the answer over Z(kn) into Z(n). A correct answer is assured because

$$(i\ j \pmod{kn}) \equiv i\ j \pmod{n}$$

For example, assume that an exponentiation subroutine is to be written using a thousand bit multiplication subroutine. Although n may be less than 2^{1000}, it is only necessary to reduce products to one thousand bits in order to prevent intermediate values from getting too large for the multiplication routine. Only to compute the final value must complete reduction to a value in Z(n) be performed.

Multiplication of integers in Z(n) consists of two operations, integer multiplication and reduction over Z(n). Knuth [4] gives an excellent presentation of a plethora of multiplication algorithms that have been developed for big numbers. Selection of a multiplication algorithm is dependent on the size of the numbers to be multiplied, the needed flexibility in the system, and the hardware available for the implementation. Knuth also describes a division algorithm which is easily converted to reduce integers over Z(n) by ignoring the evaluation of the quotient and

only computing the remainder.

An alternate approach to computing the remainder is to approximate the quotient by multiplying by the reciprocal of n. This guess at the quotient can then be multiplied by n and the product subtracted from the integer to be reduced. Since only multiples of n are subtracted, the value in Z(n) is preserved. Here the concept of approximate reduction is particularly useful since the approximate quotient (sometimes being too small) may not cause the subtraction to yield a value in Z(n). The final answer is then computed from the approximately reduced answer by successive subtraction or use of a traditional division algorithm.

The choice of method for the final reduction of an approximately reduced answer depends on how near n is to the basic maximum integer of the system. If fixed size integers are used and the modulus is guaranteed to be within a few bits of the maximum integer size then successive subtraction is adequate. However if the modulus may be significantly smaller than the maximum integer (or if variable length integers are used) subtraction may be highly inefficient and some division algorithm must be used.

It is desirable to have the approximate quotient less than the actual value when it is not equal. Otherwise subtraction yields a negative value and extra addition is necessary to obtain a positive integer.

The reciprocal of n is constant and therefore need only be computed once per conversation. The following algorithm computes such approximate quotients. Fixed size integers are used; B-1 is the maximum integer. Subscripts "$_L$" and "$_H$" denote low and high order portions, respectively. The following identity defines the use of the subscripts.

$$x_H x_L = B\, x_H + x_L$$

Algorithm 3. To reduce $a_H a_L$ (mod n).

1.	$R_H R_L \leftarrow (B^2)+n$	{compute 1/n}
2.	$Q_H Q_L \leftarrow ((a_H)\,(R_H R_L))+B$	{approximate quotient}
3.	$P_H P_L \leftarrow n\,Q_L$	{multiple of n to subtract}
4.	$a_H a_L \leftarrow a_H a_L - P_H P_L$	{reduced value of a}
5.	if $a_H \neq 0$ then goto 2	{repeat until high part is 0}
6.	output a_L	{reduced value (mod n)}

If $a_H a_L$ is less than n^2 then Q_H is always zero. The branch at step 5 is rarely taken since the approximate quotient is usually very close to the actual value. Note that step 1 need only be done once for each value of n. Also the division by B should be made trivial by choosing B such that shifting by some number of words divides by B (i.e. on a 32 bit computer choose B to be, say, 2^{320} so that a ten word shift divides by B).

In summary, the following arithmetic functions are necessary to encrypt or decrypt fixed size integers (of maximum value B-1). Each operation is represented as a mapping from the cross product of two sets into a third. B^2 denotes a double length integer (with maximum value B^2-1), etc.

Figure 2. Arithmetic functions needed for the RSA method.

Subtraction:	$B^2 \times B^2 \rightarrow B^2$	
Multiplication:	$B \times B \rightarrow B^2$	
	$B \times B^2 \rightarrow B^3$	
Divison:	$B \times B^3 \rightarrow B^2$	(for 1/n only)

Prime construction. An individual wishing to participate in public-key communication must construct the public encryption information <n,e>. The magnitude of the modulus (n) is determined by the difficulty of breaking the code chosen by the constructor of the parameters. The following table is taken from [11] and is useful in choosing the magnitude of n. The number of operations computed for each value of n is based on the best known factoring algorithm for

large integers.

Figure 3. Effort required to factor n.

$\log_{10} n$	Operations	Remarks
50	1.4×10^{10}	
100	2.3×10^{15}	(At the limits of current technology)
200	1.2×10^{23}	(Beyond current technology)
400	2.7×10^{34}	(Requires significant advances)
800	1.3×10^{51}	

Once the magnitude of n is chosen, two primes must be randomly selected such that their product is of the chosen magnitude. The prime number theorem states that the primes near n are spaced on the average one every (ln n) integers. Thus even for large primes several hundred digits long, only a few hundred candidates need be tested before finding a prime. This selection procedure is quite time consuming (about an hour on a medium scale computer), yet it only needs to be done once.

To confound certain factoring algorithms it is desirable to have primes, p, such that p-1 has a large prime factor. Therefore the following nonsequential scan is recommended. Choose a large prime (p) by sequential scan of odd numbers such that the magnitude of p is slightly less than that of the desired prime. Now scan from 2p+1, in increments of 2p, until a prime is found. Since this prime is of the form kp+1, one less than it has a large prime factor, namely p.

Prime testing is (fortunately) much faster than prime factoring. Probabilistic algorithms [10] exist which decide with an arbitrarily small uncertainty if an input is prime. It is recommended that a combination of two methods be used. Both algorithms are probabilistic and have the same conceptual basis. Tests exist which can reliably identify composites and which never mistake a prime for a composite; however, some composites are mistaken for primes. The probabilistic algorithm consists of making many such (independent) tests, declaring the input composite when any one test yields that result. If a number passes numerous tests as prime, it

is assumed to be prime. If the test mistakes a composite for being prime with probability P, then by using k tests the probability that the algorithm will incorrectly declare a composite to be prime is P^{-k}.

The first method is based on the Euler-Fermat theorem previously mentioned. Since for prime p, $\phi(p) = p-1$, the theorem states

$$p \text{ prime } \& \ (a,p) = 1 \ \Rightarrow \ a^{p-1} \equiv 1 \pmod{p}$$

When p is prime (a,p) is one for all $a < p$. Thus the test for p prime is

$$\text{Prime}(p) \ \leftrightarrow \ (\forall a < p)[\ a^{p-1} \equiv 1 \pmod{p}\ ]$$

In practice only a few values are chosen from Z(p), not all as indicated above. Only a small number of composites pass this test even a few times. Thus it is recommended that approximately one hundred tests be made to reliably conclude that a selected number is prime. Experience indicates that choosing a = 3 will identify virtually all composites.

This test is very attractive since the computation required is simple. However, composites are known which will pass this test as primes. The Carmichael numbers ($561 = 3 \cdot 11 \cdot 17$ is the smallest) pass the test for all values in Z(p). A more exhaustive test has been devised by Solovay and Strassen [12] which identifies even Carmichael numbers as being composite. The test states that

$$\text{Prime}(p) \ \leftrightarrow \ (\forall a < p)[\ (a,p) = 1 \ \ \& \ \ a^{(p-1)+2} \equiv J(a,p) \pmod{p}\ ]$$

where J(a,p) denotes the Jacobi symbol. The Jacobi symbol is defined as

$J(a,p) = 1$ when $x^2 \equiv a \pmod{p}$ has a solution in Z(p).

$J(a,p) = -1$ when $x^2 \equiv a \pmod{p}$ has no solution in Z(p).

Solovay and Strassen prove that at least half of the elements of Z(p) for any composite p will cause the test to identify p as composite. Therefore making k tests yields an answer that is wrong with probability 2^{-k} when claiming the input is prime (and always correct when claiming it is composite).

The following algorithm is recommended for testing a number p for primality. It combines the two methods discussed above, the Euler-Fermat test being used as a quick indication of

compositeness, with the Solovay-Strassen test serving to verify that numbers passing the first quick test are indeed prime. The algorithm includes a standard formula for computing J(a,p).

Algorithm 4. To test if p is prime.

1. if $3^{p-1} \not\equiv 1 \pmod{p}$ then output FALSE
2. do steps 4 through 6, 100 times
3. output TRUE
4. a ← random({2, ... ,p-1})
5. if gcd(a,p) ≠ 1 then output FALSE
6. if $J(a,p) \not\equiv a^{(p-1)+2} \pmod{p}$ then output FALSE

Algorithm 5. To compute J(a,b).

1. if a = 1 then output 1
2. if even(a) then output $J(a+2,b)\ (-1)^{(b^2-1)+8}$
3. output $J(b \pmod{a}, a)\ (-1)^{(a-1)(b-1)+4}$

Parameter selection. Given two primes whose product is of sufficient magnitude to assure the security of the code, the public information is chosen as follows. The modulus is computed as

$$n = p\,q$$

where p and q are the two primes chosen by the method described above. It is now easy to compute $\phi(n)$ since the factors of n are known.

$$\phi(n) = (p-1)\,(q-1)$$

The secret information, d, is now chosen to be a large integer in $Z(\phi(n))$. Only a d relatively prime to $\phi(n)$ is acceptable since it must have a multiplicative inverse in $Z(\phi(n))$. The inverse of d is denoted e and together with n forms the complete public information, $\langle n,e \rangle$. Computing the gcd of d and $\phi(n)$ determines quickly if they are relatively prime and has the added side effect of also computing the multiplicative inverse of d at the same time. It has been shown that the

gcd function is computable in O(log n) time [3], therefore performing this computation is reasonable. The following algorithm computes both the gcd and multiplicative inverse (e).

Algorithm 6. To compute gcd(d,f), d^{-1} (mod f).

1. (A0,A1,A2) ← (f,1,0); (B0,B1,B2) ← (d,0,1)
2. if B0 = 0 then output A0 {No inverse exists}
3. if B0 = 1 then output B0,B2
4. Q ← ⌊ A0 ÷ B0 ⌋
5. (T0,T1,T2) ← (B0,B1,B2)
6. (B0,B1,B2) ← (A0,A1,A2) - (Q B0,Q B1,Q B2)
7. (A0,A1,A2) ← (T0,T1,T2)
8. goto 2

Thus parameter selection consists of three stages. First the degree of security demanded for the application is estimated and an appropriate magnitude for the modulus is chosen. Next two large prime numbers are chosen whose product (which will be n) is just slightly greater than the minimum magnitude decided upon. Should the product greatly exceed this magnitude, encryption and decryption will take longer than necessary. Finally from the pair of primes and a randomly chosen secret number, the public information is computed.

Tests for the constructed parameters. It has been shown how to create parameters for the RSA trap door one-way function so that decryption will always recover the original message. Further constraints must be placed on these parameters so that all attacks known (to the author) are foiled. These extra tests reject some values of <n,e> as being too easy to break. When a value is rejected, the selection algorithm must go back to the last place where random information was introduced into the computation of the rejected parameter. The determination of the magnitude of n is application dependent and shall be assumed to be chosen safely.

Both primes constituting the modulus (n) must not have any peculiarities that can be taken

advantage of by factoring algorithms. It is suggested that both p-1 and q-1 have large prime divisors. Furthermore neither should be a Fermat or Mercene number since these special primes are well studied and the resulting product therefore seems more likely to be factorable.

The two primes should differ by a few orders of magnitude. If $p \simeq q$, then $2\sqrt{n}$ is a good approximation of p+q. Knowing p+q immediately gives $\phi(n)$ since

$$\phi(n) = (p - 1)(q - 1) = n + 1 - (p + q)$$

The average of p and q is only near $\sqrt{n}$ when p and q are nearly equal. Thus calling the large of the primes p, a large value of p+q is recommended. Obviously neither p nor q should be small. The reader is referred to Pollard [9] and Knuth [4] for additional approaches to the factoring of large integers.

The encryption and decryption parameters should both be large. If $e < \log_2 n$ then small messages will not be disguised by the modulo reduction; that is for small message m,

$$m^e \pmod n = m^e$$

and the code word is breakable by brute force. The selection of such a small inverse is very unlikely, but nevertheless must be excluded when the message, m, can be small.

When the secret exponent, d, is smaller than $(\log_2 n)$ the code is easily broken. Should the enemy assume that d is small, a random search will quickly determine its value. Also, if a message is discovered that encodes into a very small code word, c, such that

$$c^d \pmod n = c^d$$

then by computing the powers of c the enemy will soon find one that matched the original message. Counting how many multiplications were needed gives the secret information, d.

The tests are summarized below giving recommended limits for different levels of security. High security tests should ensure that no reasonable attack can be made using existing theory and foreseeable technology. Medium security tests give a much less constrained cryptosystem within which encryption and decryption are several times faster than high security. However, it seems within the reaches of future technology to attempt to break such a code. Certainly no known attack is possible today.

Figure 4. Summary of Tests.

Test	High security	Medium	
$n > \alpha$	$\alpha=10^{200}$	$\alpha=10^{100}$	
$p+q > \alpha$	$\alpha=10^{5}$	$\alpha=10^{3}$	(Assuming p > q)
$p+q < \alpha$	$\alpha=10^{20}$	$\alpha=10^{10}$	
$[d\|(p-1)] > \alpha$	$\alpha=10^{80}$	$\alpha=10^{30}$	
$[d\|(q-1)] > \alpha$	$\alpha=10^{80}$	$\alpha=10^{30}$	
$\alpha < d < n-\alpha$	$\alpha=10^{20}\log_2 n$	$\alpha=10^{10}\log_2 n$	
$\alpha < e < n-\alpha$	$\alpha=10^{20}\log_2 n$	$\alpha=10^{10}\log_2 n$	

C. Details of the Method.

Signatures. Messages sent using the RSA method, as well as any other method based on trap door one-way permutations, can be signed by the sender [1]. A signed message has all the characteristics of a signed letter from an informational point of view. It is the advent of this ability to sign digital correspondence which allows electronic communication systems to begin to replace existing postal communication systems. In addition to containing information (the message), a signed transmission must have these properties:

(i) Only the authorized recipient can decode the message.

(ii) The authorized recipient is assured that the transmission is not forged.

(iii) The authorized recipient can prove to a third party who sent the transmission

(iv) The proof in (iii) reveals no secret information.

Signed messages can be sent using the RSA method. The principle is that the two communicants compose their encryption and decryption permutations, in such a way that each communicant can only compute one direction of the new composite permutation. Since each pair of communicants has a unique composite permutation, the originator and intended recipient are clearly defined. To send a signed message requires a decryption permutation, therefore only one communicant could be responsible for a message which has had a particular decryption permutation applied to it.

Consider the case of a communicant S (with encryption permutation E_S and decryption permutation D_S) sending a signed message, M, to R (with E_R and D_R). The transmitted code word, T, is computed to be

$$T = E_R(D_S(M))$$

Only R can decypher T because E_R was used in creating it, therefore D_R must be applied to the transmission to make any sense out of it. Similarly, S is obviously responsible for the

transmission because D_S was applied to the message M.

When R receives the transmission, T, the original message is recovered by computing

$$M = E_S(D_R(T)) = E_S(D_R(E_R(D_S(M)))) = E_S(D_S(M)) = M$$

From the above it is clear that R must somehow learn that S sent the message in order to know that D_S was used. If this information is not somehow implicitly known, then S must include the plaintext name ("S") with the transmission to allow R to apply the correct encryption permutation.

An independent third party can be convinced by R that S was responsible for the transmission from E_S, $D_S(M)$, and M. It is easy for R to derive $D_S(M)$ from the transmission, T, by computing $D_R(T)$.

$$D_R(T) = D_R(E_R(D_S(M))) = D_S(M)$$

The third party knows that M was sent by S because both $D_S(M)$ and M are known by R who does not know D_S. Unless the message adds additional information, however, the third party does not know that the message was intended for R.

Signatures are only provable of a specific origin so long as the signer maintains the secrecy of his decryption function. Once the secret is lost, the validity of all signed messages, past and present, is questionable. A signer can disavow a signed message by intentionally revealing his secret information to an enemy. This problem is not only characteristic of digital signatures; people lose signature stamps, credit cards, etc.

The solution is for the recipient of a critical message to submit it to a judge secretly. After allowing a reasonable amount of time for the signer to report any loss of his secret information, the signer is informed that his signature is being tested. Since the judge has had the evidence since before the secret information could reasonably be claimed to have been lost, the signer is clearly responsible for the message.

RSA signatures. The RSA method has a special problem implementing signatures due to the nature of the trap door one-way permutations used. Say a given pair of communicants, R and S, have public parameters, $\langle n_R, e_R \rangle$ and $\langle n_S, e_S \rangle$ and wish to communicate using signatures. If n_S

> n_R then it is possible that $D_S(M) > n_R$ making $E_R(D_S(M))$ not computable. Message alteration, discussed further on in this section, is one solution - reword the message until $D_S(M)$ fits into E_R. Rivest, Shamir, and Adleman [11] suggest several other approaches to the problem, none of which are completely satisfactory.

A solution to this problem and to the problem of determining the intended recipient is suggested here. First the general approach to forming signatures is discussed. Then this method is applied to the problems just mentioned.

Signatures are formed by composing the permutations in the opposite order. Thus,

$$T = D_S(E_R(M))$$

The recipient decodes this signed message by computing

$$M = D_R(E_S(T))$$

Using this method, the third party is convinced of the identity of the sender from E_S, E_R, T, and M. The third party knows that S is responsible for the transmission because only S could have computed T such that

$$E_S(T) = E_R(M)$$

Furthermore, the third party knows that R was the intended recipient because E_R is inside the expression D_S was applied to.

If it is not necessary for the intended recipient to be indicated by the signature, a combination of the two signature methods works.

$$T = E_R(D_S(M)) \qquad \text{when } n_S < n_R$$

$$T = D_S(E_R(M)) \qquad \text{when } n_S > n_R$$

Since R knows both n_R and n_S, he can decide in which order the permutations were composed and therefore select how to decypher the message.

When the signature must indicate the intended recipient the second signature method should be used. Now the RSA signatures will not work when $n_S < n_R$. In this case some solution such as those presented in [11] must be employed. One answer is to require all communicants within a system to have two <n,e> pairs. A system-wide constant is selected and it is required that

each communicant have one n greater than the prescribed constant and one n smaller. Then to send a signed message, the sender uses his larger decryption permutation, applying it to the message as encrypted by the recipient's smaller permutation.

Key representation. The key is the public information associated with each instance of the RSA public-key cryptosystem, <n,e>. To these parameters additional information may be added as a convenience or as a precaution against abuse of the key.

When the Public File sends key information to communicants, the name of the communicant whose key was requested must be bound into that message. Such a transmission from the Public File might look like

$$D_{PF}(E_C(\text{"X, } \langle n_X, e_X \rangle\text{"})$$

when communicant C seeks information on communincant X. Should the name, "X", not be included in the same signed portion of the message, impersonation is allowed. X could get the signed parameter specification for X from the Public File. Then when some communicant asks the Public File for a key, X could intercept the communication and send back the Public File's response for X's key. Since it is a valid response the unsuspecting communicant is fooled and X can then impersonate the communicant whose key was requested.

One way of computing the encryption function previously discussed involved approximating division by n using the reciprocal of n. It was pointed out that the calculation of the reciprocal need only be done once for each n. The Public File could compute the reciprocal of n and transmit that, along with n and e, as a part of the key information. This way, no division need ever be done by a communicant. If encryption is to be computed by an inexpensive piece of hardware, eliminating the need for a division function is clearly advantageous.

The following format is recommended for keys. The sizes of the fields will of course depend on the particular application.

Figure 5. General Key Representation

$< T, n, e, n^{-1} >$

T	Communicant identifier
n	Encryption modulus
e	Encryption exponent
n^{-1}	1/n in a standard form

Message alteration. It is sometimes necessary to manipulate a plaintext message before encrypting it. As mentioned previously, short messages or messages whose range of contents might be guessed by the enemy should not be sent. Furthermore, message alteration is a good way of avoiding signature reblocking problems as discussed above. Finally, this technique solves the problem of possibly transmitting the same message more than once.

Time stamping and sequence numbering of transmissions are forms of message alteration since they make messages unique and help to obscure simple messages. However, although these are good techniques, they alone are not a sufficient form of alteration. The enemy knows the time of day and by counting transmissions can deduce sequence numbers. Therefore a sophisticated enemy could predict the form of the altered message and a simple message would no longer be obscured.

A good and simple solution is to add a small random field to each message block before it is encyphered and transmitted (in addition to sequencing and time stamping). A field of ten characters is sufficient to befuddle most enemies, and one of twenty characters should effectively mask even the simplest of messages from a powerful enemy. Such a small field will cause a slight data expansion from the plaintext to transmitted code, but the added security seems to warrant this small waste.

If data compression is crucial to the application, a more complicated approach can be used to help the situation. It can rather safely be assumed that long messages will probably not be

guessed by the enemy. Therefore, the random character field need only be added to short message blocks (along with, of course, some indication to the other communicant that the field is to be ignored). In this way large volumes of data can be transmitted as efficiently as possible with random fields being appended only to short blocks. If necessary, time stamping can be eliminated in favor of the more compact sequence numbering scheme.

A complete communication protocol. All of the issues previously discussed may now be combined to form a working public-key cryptosystem. Suppose S wants to send a signed message to R. Each communicant consults the Public File (PF) to obtain the other's public encryption information. Only the name "R" and a way of forming a communication link to R are known initially to S (all of which may be publicly known). The message to be sent is denoted M. Each message will contain a random field, time stamp, and sequence number. The validity of the time and sequence fields will always be checked and the authenticity of the communication will be doubted should a discrepency appear. These standard fields are not shown in the outline that follows.

First the Public File is consulted to obtain the information deposited there by R.

S to PF: E_{PF}("Send info on R. Signed, S.")

The Public File responds with the requested public information. The information is signed to assure S that it is genuine. If the signature functions do not compose as shown here, they are applied in reverse order and then S in turn will decypher as necessary.

PF to S: $D_{PF}(E_S(\text{"R: } \langle n_R, e_R \rangle\text{."}))$

At this point S has enough information to compute E_R. Denoting the above transmission as $T_{PF \to S}$, S decodes the signed message by computing

$$D_S(E_{PF}(T_{PF \to S}))$$

and extracting from the plaintext numerical values for the parameters. Of course S checks that the correct name, "R", appears in the plaintext message.

Now S can open communication with R and send a signed transmission.

S to R: "R: This is S."

S to R: $D_S(E_R(M))$

The first message informs R of who is attempting to make contact. Extracting the name, "S", R initiates communication with the Public File. In a dialogue exactly like S had above with the Public File, R learns the information necessary to compute E_S. Now R is in a position to read the message. Again, if the functions did not compose as shown above, they would be applied in the opposite order and then R would in turn make the appropriate adjustments in the following computation.

$$M = D_R(E_S(T_{S \to R}))$$

In this way R learns the original message sent by S. Two way communication can proceed directly from this point since both communicants now know each others' encryption function.

Throughout this entire interchange, time and sequence fields are automatically being checked. Since transmission difficulties could cause these checks to find errors, rather than completely terminating the communication it is better to request a retransmission, giving the expected time and sequence number in plaintext. Such a retransmission request must be uniquely identifiable from coded text to be recognized as such by the original sender.

III. Implementing the RSA Method.

A. Certificates.

Motivation. The major difficulty in initiating a public-key communication is reliably determining the correct encryption key to use. Since these keys are typically transmitted over public channels (if secure channels were available, traditional cryptographic methods could be used) the origin of key transmissions is often questionable.

One solution is for the communicants to exchange keys before commencing enciphered communication. If the channel is known to reliably connect the authorized communicants then this technique is adequate. However if there is any chance that an enemy could transmit messages then this approach is not secure. If an enemy can transmit a key then impersonation is possible. Without other information, the enemy's key is as genuine as any other.

Another approach that works even when an enemy can transmit messages is for all communicants to get keys from the Public File. Prior to communicating with another communicant, the Public File is contacted and requested to send the appropriate key information. Each individual has a name in the system by which he is referenced in the Public File. Once two communicants have gotten each others' keys from the Public File they can securely communicate. The Public File digitally signs all of its transmissions so that enemy impersonation of the Public File is precluded.

Yet this solution is far from ideal. Continually referencing the Public File is a nuisance. When a communicant is initially contacted he must suspend that communication, get the appropriate key from the Public File, and then resume the original communication. Thus either the communicant must use two communication lines at once or break and then reinitiate a communication link. Clearly the exchange of keys is much more convenient in this regard.

Furthermore there are many new problems caused by the introduction of the Public File. If it is frequently used it will need to be a very large and complex system. Securely updating such a large system will be difficult. The communications equipment will be very expensive

since is must be secure against tampering.

Certificates. The use of certificates allows key information to be obtained as reliably as if it were from the Public File without ever making contact with the Public File. There is a certificate for each communicant in the system. Each certificate can only be created by the Public File and contains a name and key information pair. Communicants can check that a certificate was created by the Public File. Communicants convey their key information to others simply by sending their certificates.

Deducing the name to key information binding from a certificate is referred to as *reading* the certificate. Proving that it was generated by the Public File is referred to as *verifying* the certificate.

A certificate is like a signed message from the Public File associating key information to a name. For a certificate to convince a communicant of another's key information it must have the following properties:

(i) Any communicant can read a certificate and verify that the Public File created it.

(ii) Only the Public File can easily create or modify a certificate.

(iii) The Public File only creates certificates with names of communicants and their associated keys.

Property (ii) uses the notion of easy creation of certificates because property (i) precludes the possibility that a digital certificate is unforgeable. Since any communicant can read and verify certificates, forgery is achieved by trying to read all possible digital certificates until the correct result is gotten. The number of possible certificates must be large enough to make such an attempt prohibitively time-consuming.

The Method. A *certificate* is an ordered triple containing an *authenticator*, key information, and a plaintext name. The certificate for the communicant U with encryption function E_U is denoted

$$< A_U, E_U(\cdot), \text{"U"} >$$

($E_U(\cdot)$ denotes a description of the function E_U; "U" denotes the plaintext name).

The authenticator, A_U, serves to bind together the other information in the certificate. Furthermore only the Public File can compute authenticators although any communicant can verify them. It is suggested that the authenticator be computed by the Public File as

$$A_U = D_{PF}(E_U("U"))$$

(D_{PF} denotes the inverse to the encryption function of the Public File). Since D_{PF} is used in the construction of authenticators only the Public File can create them. The other two items in the certificate are known to be bound together by the Public File because they both appear inside the application of the secret Public File decryption function.

Certificates can be read by anyone since the key information and communicant name appear as plaintext. Any communicant knowing the Public File encryption function (which is public knowledge) can verify that the authenticator binds the name and key information. To check a certificate of the form

$$< A, E(\cdot), N >$$

it is only necessary to verify that

$$E_{PF}(A) = E(N)$$

The new role of the Public File. The Public File is no longer involved in the initiation of communication between any pair of communicants. Only two basic responsibilities of the Public File remain. First, names must be unique throughout the system. Certificates are useless if they are available from the Public File with arbitrary names bound to functions. The Public File must only issue one certificate for a given name in the life of the system. Secondly, the Public File must maintain the secrecy of its inverse function.

New communicants must visit the Public File once before they can begin communication within the system. Each new communicant presents to the Public File a name and encryption function information. After checking that the name is unique in the system, the Public File computes the certificate. The communicant is given his certificate and the Public File encryption function algorithm enabling him to verify certificates. No further interaction between the

communicant and Public File is ever necessary.

The introduction of the certification method has neither increased nor decreased the powers of the Public File. An enemy subverting the Public File can cause bogus certificates to be issued. Yet when the Public File is contacted (i.e. when not using certificates) a similar enemy subversion could cause falsified information to be sent in response to a request for key information. Thus the use of certificates does not introduce any new weaknesses into the system.

Lost decryption functions. The integrity of a public-key communication system is seriously jeopardized by the loss of a secret decryption function to the enemy. With or without certification methods enormous damage can be done by the enemy before the loss is discovered. Even when the loss is reported, the confidentiality of previous correspondence is necessarily violated. It is more difficult to recover from such a loss when certificates are used although no public-key scheme can in any sense recover well under such circumstances.

When the Public File is always contacted to get key information, a communicant who has lost his decryption function can have a new encryption function placed in the Public File. From that time on a secure key is given out and the old decryption function is useless, except for decyphering old messages.

When certificates are used such a cancellation of existing functions is not possible because a certificate, once created, is spread throughout the system as public knowledge and is not destroyable. The problem is similar to that of preventing misuse of lost credit cards. Three solutions are given here. For any given system, one of the approaches suggested should adequately handle the problem of lost secret information.

One solution to the problem is to inform all communicants in the system of any lost decryption functions. In this way every communicant can maintain a list of names of cerficates to be ignored. Unless the system is of enormous size, this solution is adequate since the lost decryption functions are presumably quite rare. If the communicants use machines which are nodes on a large network, then only these nodes need to be informed of bad certificates. Each

certificate can be checked with the local node and no longer must each communicant be explicitly informed of every lost decryption function.

Another approach is to put expiration dates on certificates. This would mean that new certificates would have to be computed and distributed periodically by the Public File, increasing the amount of communicant interaction with the Public File and therefore reducing the advantages gained by using certificates. The expiration date must of course be included in the authenticator as well as appear in plaintext.

A third approach is to allow certificates to be double checked. In the case of a suspicious communication, the Public File could be consulted to check if the associated certificate has been cancelled due to a security leak. When a communicant loses his decryption function he immediately notifies the Public File and those with whom he regularly communicates. Any other communicants contacted by the enemy using the stolen decryption function will hopefully find the communication suspicious and request verification from the Public File.

Evaluation. The method of certification presented here is an adequate replacement for the method of requesting information from the Public File. Certificates do not increase the responsibilities of the Public File nor do they increase its power to do damage when subverted. In practice, the freedom of not having to contact the Public File is a considerable advantage.

Certification does not work well when secret key information is disclosed to an enemy. Public File requesting is sometimes better when secret key loss is common, although the damage done is tremendous using any method. Several techniques do exist to recover from the loss of a decryption function when certificates are used. Depending on the size and organization of the system, one of these enhancements to the basic method should reasonably suffice to maintain system integrity.

B. A Suggested RSA Implementation.

This section consists of a detailed specification for a large scale communication system using the RSA method for maintaining message privacy. Many details are suggested to help insure the secure operation of the system. An overall evaluation of the system is included to indicate the strengths and weaknesses of the RSA method in an application.

System Overview. The system presented here allows secure communication between individuals on a computer network via an electronic mail system. Message encryption and decryption is done at the user terminal because it is too dangerous to ever enter a decryption key into a multi-user system. Computers on the network would maintain a library of certificates as well as handle message transmission between users. A Public File would need to exist to create certificates.

The Terminal. The system is designed around special-purpose terminals that do the message encryption and decryption, as well as handle the standard message protocols. The terminals should have sealed cabinets and always be kept in a restricted access area to prevent tampering. Each should have a CRT screen so that messages can be displayed temporarily and never need appear on paper. In addition, a hard-copy device would be useful to record messages that cannot be memorized. The terminal contains specialized high-speed logic to multiply large integers as well as a microcomputer to handle communication protocols with the computer system.

The Card. Each terminal has a magnetic card reader to be used to enter encryption and decryption information. A user identification name must appear on the card to insure that the originator of a given message is always reliably known. The secret information on the card will not be stored in plaintext as is the public information. Each communicant knows a short password which must be given to access the secret information contained on the card. Using some standard encryption algorithm, the password acting as a key decrypts the secret information from the encrypted form stored on the card. This guarantees that loss of the card does not immediately give away the ability to decode messages, giving the authorized

communicant some time to destroy as much encrypted information as possible and to warn the system that an attack may occur.

On the card is stored the communicant identification name, the public information for the code, <n,e>, an encrypted form of the secret decoding information, d, and a standard representation of 1/n. It is required by the system that

$$10^{397} < n < 10^{400}$$

to assure that all n are roughly the same order of magnitude. Furthermore the computers in the network will use $n > 10^{400}$ so that signature is always possible with a consistent order of composition. These 400 digit numbers can be stored on the cards as 200 characters. Therefore each item on the card occupies a 200 character field.

Message format. All messages sent between terminals and computers are of the same format, be they encrypted or plaintext. The basic format consists of a byte count, a sequence number, and a time field, followed by data. Long messages are broken into blocks no bigger than 256 bytes. If the data is encrypted, the first ten characters represent the random field and are to be ignored after decryption.

Should either the terminal or computer detect a discrepency in the sequence or time fields, retransmission is immediately requested. A retransmission request is uniquely distinguishable from a message transmission (by setting the sequence field to 0) and must always be immediately satisfied. Should the disparity continue, the communication is terminated and the connection must subsequently be reinitiated.

Communication protocols. To initiate a session at a terminal, the communicant must first plug his magnetic card into the terminal. Next the password is entered by the communicant and then checked by the terminal. The password is checked by determining that the resulting secret information does indeed invert the public encryption function as stored on the card. Once the password is correctly given, a list of any queued messages is requested from the computer network and displayed on the screen. This request is signed to insure that another communicant does not get the list of queued messages. The communicant may choose to either

receive a queued message or transmit a message to another communicant.

To receive a message, the communicant need only select it from the list of queued messages previously presented. The terminal then requests the computer network to send the certificate for the communicant who sent the message. Coded text is passed from the network to the terminal where it is decoded and displayed on the CRT screen. The communicant can request hard-copy or request that the message remain queued and not be deleted after it is read.

To transmit a message, the communicant first enters the identification name of the communicant to receive the message. The terminal gets the appropriate certificate from the computer network and begins to send signed text as it is entered by the communicant. Editing within a line is permitted, but any more extensive editing abilities would probably represent too great a memory demand on the terminal.

Perhaps in addition a conversational mode would be available. In this mode two communicants both logged on at the same time could interconnect their terminals through the network. Encrypted information would pass between the terminals via the computer network.

When the communicant is finished, the terminal performs a clear sequence. All writable memory is zeroed and the CRT screen is cleared.

The Public File. The Public File does not have to be a computer system but rather a terminal with some special software. To create a certificate an authorized administrator would log onto the network at the special terminal by giving the necessary password and presenting the Public File magnetic card. The encoded file of existing users would be checked to assure that the name requested is unique. If unique the new name is added to the encoded file.

To create the certificate, the system administrator would plug the card of the new communicant into the terminal. The card would be read to check the name and get the encryption information. From this information the certificate is computable since this special terminal knows the Public File decryption function. Once computed the certificate is broadcast to the computer network where it is permanently stored. Since the decryption information on the card is coded, the Public File never learns any communicant's secret information.

Evaluation. The system described above seems quite workable although rather expensive. Data rates of at least 120 characters per second should be realizable using bipolar logic to do the main computation. The physical security of the terminals is one of the few potential weaknesses. Given that the terminals are sufficiently secure, the only point of attack remaining would be the computer network.

In a system of several computers hooked up in a network, they could all keep each other honest. In idle time a computer could contact another, impersonating a communicant at a terminal, and request a certificate. The received information would be checked against the local copy of that same information with any discrepency being immediately reported to a system administrator.

Another addition to the system would make it impervious to the attack of even an enemy within the computer network. Each communicant could give copies of his magnetic card to other fellow communicants. Then instead of requesting public encryption information from the computer system, it could be read in from a magnetic card. In this way as long as the physical security of the terminals and cards are maintained, message transmission is secure.

C. Future work with the RSA method.

Unbreakability proofs. Although convincing evidence indicating that the RSA method is unbreakable has been presented, there still exists the potential threat of the discovery of a useful attack strategy. Of course the lack of proof against attack plagues cryptographic methods in general. However public-key methods seem more vulnerable since the enemy may very well know the encryption function and therefore can generate arbitrary amounts of error-free ciphertext.

As study of the problem of factoring large integers and study of the RSA method in particular continue, confidence in the method will increase. Assuming no attacks will be discovered and no unbreakability proofs will be formulated, only continued effort to break the method can help to certify it.

Hardware implementations of the RSA method. The hardware of most general purpose computers seems insufficient to compute exponentiation modulo a large integer quickly enough to support fast data transmission rates. Therefore special hardware must be developed to do these computations. Speed of multiplication is crucial since thousands of multiplications are necessary to encrypt a message of a few hundred characters. Even existing bipolar logic is hard pressed to raise thousand bit numbers to large powers quickly.

Direct combinational formation of the product of thousand bit integers seems technologically infeasible. Serial multipliers seem to be the best existing solution; further work on their specialization to the exponentiation problem could be done. Other approaches to the problem of exponentiating modulo a large integer should certainly be studied.

As mentioned earlier, multiplication modulo a large integer can be done with three multiprecision multiplications. These three operations could be pipelined, with the result of one multiplication immediately entering the input of the next. Once started up the overall speed to multiply modulo an integer would be the speed of the slowest multiplication in the pipeline.

An exponentiation method was given based on the decomposition of the exponent into the

sum of powers of two. Other decompositions are of course possible, some of which might require fewer multiplications. Study of the general problem of efficient decomposition of exponents may yield a class of efficiently computable exponents from which encryption and decryption exponents may be chosen.

Construction of a communication system using the RSA method. The final test of any communication system is its implementation. A complete understanding of all the intricacies of the method is not possible until a full implementation exists. A prototype system could be constructed and put into operation in an environment of only moderate security or even one where encryption was not really necessary. Once in operation, the team who (conscientiously) constructed the system would try to break it both analytically and otherwise. Successful attacks would indicate weaknesses deserving further attention.

A prototype system would also indicate the expense involved in special hardware and implementation time. Such a system would have to be flexible enough to handle situations such as system crashes and lost keys. The user community would identify any burdens placed on communicants such as the system being too slow, passwords too hard to remember, etc.

Other new ideas could also be tested in such a prototype system. For example, the encryption functions could be incorporated into the operating system. Thus, to log in a user would have to decrypt challenges sent to his terminal. Files could exist in the system in encrypted format; information to and from the files would be routed through the user terminal which would do the necessary cryptography.

Most importantly the implementation and use of such a system will suggest new techniques and ideas to those who work with it. Public-key cryptography is a powerful tool to manipulate information in a public environment. Its potential applications in information systems of all kinds are boundless.

Acknowledgements. I thank Len Adleman, my thesis advisor, for suggesting the topic of this thesis and many helpful suggestions. I also thank Ron Rivest and Ralph Merkle for other useful discussions. Wendy Glasser was a great aide in my use of the MIT computing facilities. I thank Betsy Gershun for her help during the final editing of the manuscript.

References.

[1] Diffie, W. and Hellman, M. New Directtions in Cryptography. ***IEEE Transactions on Information Theory IT-22***, 6 (Nov. 1976), 644-654.

[2] Hardy, G.H. and Wright, E.M. ***An Introduction to the Theory of Numbers.*** Oxford Press, New York, 1960.

[3] Knuth, D.E. ***The Art of Computer Programming.*** Fundamental Algorithms (Vol. 1). Addison Wesley, Reading, Mass., 1969.

[4] Knuth, D.E. ***The Art of Computer Programming.*** Seminumerical Algorithms (Vol. 2). Addison Wesley, Reading, Mass., 1969.

[5] Merkle, R.C. Secure Communications over Insecure Channels. ***submitted to the Communications of the ACM.***

[6] Merkle, R.C. and Hellman, M.E. Hiding Information and Signatures in Trap Door Kanpsacks. ***submitted to the IEEE Transactions on Information Theory.***

[7] Miller, G.L. Riemann's Hypothesis and Tests for Primality. ***Proceedings of the Seventh Annual ACM Symposium on the Theory of Computing.*** 234-239.

[8] Niven, I. and Zuckerman, H.S. ***An Introduction to the Theory of Numbers.*** John Wiley, New York, 1966.

[9] Pollard, J.M. Theorems on factorization and primality testing. ***Proceedings of the Cambridge Phil. Society (1974).*** 521-528.

[10] Rabin, M.O. Probabilistic Algorithms, in ***Algorithms and Complexity,*** Traub, J.F. Ed. Academic Press, New York, 1976.

[11] Rivest, R.L., Shamir, A., and Adleman, L.M. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. ***Communications of the ACM 21*** 2 (Feb. 1978), 120-126.

[12] Solovay, R. and Strassen, V. A Fast Monte-Carlo Test for Primality. ***SIAM Journal on Computing*** (Mar. 1977), 84-85.