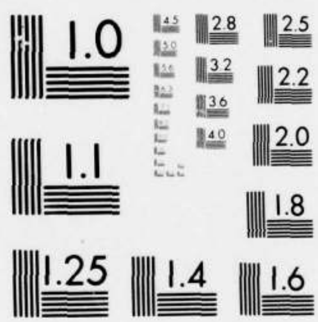
AD-A039 036 MASSACHUSETTS INST OF TECH CAMBRIDGE LAB FOR COMPUTE--ETC F/G 17/2
ON DIGITAL SIGNATURES AND PUBLIC-KEY CRYPTOSYSTEMS.(U)
APR 77 R L RIVEST, A SHAMIR, L ADLEMAN N00014-67-A-0204-0063
UNCLASSIFIED MIT/LCS/TM-82 NL

1 OF 1
AD A039036

END
DATE FILMED
5-77



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

LABORATORY FOR COMPUTER SCIENCE
(formerly Project MAC)

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

MIT/LCS/TM-82

ON DIGITAL SIGNATURES AND PUBLIC-KEY CRYPTOSYSTEMS

RONALD L. RIVEST
ADI SHAMIR
LEN ADLEMAN

THIS REPORT WAS PREPARED WITH THE SUPPORT OF NATIONAL SCIENCE FOUNDATION GRANT No. MCS76-14294 AND IN PART BY THE OFFICE OF NAVAL RESEARCH UNDER CONTRACT No. N00014-67-A-0204-0063

APRIL 1977

545 TECHNOLOGY SQUARE, CAMBRIDGE, MASSACHUSETTS 02139

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 14 MIT/LCS/TM-82	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): 6 On Digital Signatures and Public-Key Cryptosystems.		5. TYPE OF REPORT & PERIOD COVERED: 9 Technical Memoranda January 1977 — April 1977
		6. PERFORMING ORG. REPORT NUMBER: MIT/LCS/TM-82
7. AUTHOR(s): 10 Ronald L. Rivest, Adi Shamir and Len Adleman		8. CONTRACT OR GRANT NUMBER(s): 15 MCS76-14294 N00014-67-A-0204-0063, NSF
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Massachusetts Institute of Technology 545 Technology Square Cambridge, Massachusetts 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 11
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research/Associate Program Director Department of the Navy/Office of Computing Activities Information Systems Program/National Sci.Foundation Arlington, Virginia 22217/Washington, D.C. 20550		12. REPORT DATE: April 1977
		13. NUMBER OF PAGES: 12
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): 12 13p.		15. SECURITY CLASS. (of this report): Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

DDC
MAY 3 1977
C

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

digital signatures	factorization
public-key cryptosystems	electronic mail
privacy	message-passing
authentication	electronic funds transfer
security	cryptography

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

We show that the single operation of raising a number to a fixed power modulo a composite modulus is sufficient to implement "digital signatures": a way of creating for a (digitized) document a recognizable, unforgeable, document-dependent digitized signature whose authenticity the signer can not later deny. An "electronic funds transfer" system or "electronic mail" system clearly could use such a scheme, since the messages must be digitized in order to be transmitted.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

409 448

MIT/LCS/TM-82

On Digital Signatures and Public-Key Cryptosystems

Ronald L. Rivest
Adi Shamir
Len Adleman

Massachusetts Institute of Technology
Laboratory for Computer Science
(formerly Project MAC)

CAMBRIDGE MASSACHUSETTS 02139

On Digital Signatures and Public-Key Cryptosystems

Ronald L. Rivest
Adi Shamir
Len Adleman

Abstract

We show that the single operation of raising a number to a fixed power modulo a composite modulus is sufficient to implement "digital signatures": a way of creating for a (digitized) document a recognizable, unforgeable, document-dependent digitized signature whose authenticity the signer can not later deny. An "electronic funds transfer" system or "electronic mail" system clearly could use such a scheme, since the messages must be digitized in order to be transmitted.

Key words and Phrases: digital signatures, public-key cryptosystems, privacy, authentication, security, factorization, electronic mail, message-passing, electronic funds transfer, cryptography.

* This research was supported by grant National Science Foundation MCS76-14294, and the Office of Naval Research grant number N00014-67-A-0204-0063

On Digital Signatures and Public-Key Cryptosystems

by R. L. Rivest, A. Shamir, and L. Adleman
MIT Laboratory for Computer Science
Cambridge, Mass. 02139
April 4, 1977 (Revised April 21, 1977)

The operation of raising a number to a fixed power modulo a composite modulus is shown to be sufficient to implement "digital signatures": a way of creating for a (digitized) document a recognizable, unforgeable, document-dependent, digitized signature whose authenticity the signer can not later deny. This scheme has obvious applications in the design of "electronic funds transfer" systems or "electronic mail" systems, since here the messages must be digitized in order to be transmitted.

I. Introduction

Our approach is to provide an implementation of a "public-key cryptosystem", an elegant concept invented by Diffie and Hellman [2]. Such a system provides digital signatures, as well as enabling enciphered communication between arbitrary pairs of people, without the necessity of agreeing on an enciphering key beforehand.

In a public-key cryptosystem each user A places in a public file an enciphering algorithm (or key) E_A . User A keeps to himself the details of the corresponding deciphering algorithm D_A which satisfies the equation

$$D_A(E_A(M)) = M \text{ , for any message } M. \qquad (1)$$

Both E_A and D_A must be efficiently computable. It is assumed that A does not compromise D_A when revealing E_A. That is, it should not be computationally feasible for an "enemy" to find an efficient way of computing D_A, given only a specification of the enciphering algorithm E_A. (Clearly a very inefficient way exists: to compute $D_A(C)$ just enumerate all possible messages M until one such that $E_A(M) = C$ is found. Then $D_A(C)=M$.) Only A will be able to compute D_A efficiently.

Whenever another user (say B) wishes to send a message M to A, he looks up E_A in the public file and then sends A the enciphered message $E_A(M)$. User A deciphers the message by computing $D_A(E_A(M))=M$. By our assumptions only user A can decipher the message $E_A(M)$ sent to him. If A wants to send a response to B he of course enciphers it using E_B, also available in the public file. Therefore no transactions between A and B are required to initiate private communication. The only "setup" required is that each user A who wishes to receive private communications must place his enciphering algorithm E_A in the public file.

If electronic message-passing systems[7] are to fully replace the existing paperwork systems for ordinary business transactions, there is an attribute of a paper message that will have to be duplicated for electronic messages: they can be "signed". More precisely, the recipient of a *"signed" message has "proof" that the message originated from the sender. This quality is stronger than mere authentication (verifying that the message when received actually came from the sender) in that the recipient of a signed document is able to convince a disinterested third party (a judge) that the signer actually sent the message. To do so, the judge must be convinced that the signed message was not forged by the recipient himself! In an ordinary authentication problem the recipient does not worry about this possibility.*

We would like to remark that an electronic, or digital, signature must be message-dependent, as well as signer-dependent. Otherwise the recipient could modify the received message by changing a few characters before showing the message-signature pair to a judge. Even worse, the recipient would be able to attach the received signature to any message whatsoever, inasmuch as electronic "cutting and pasting" of sequences of characters are entirely undetectable in the final product.

In order to implement signatures it is necessary that E_A and D_A effect permutations of the same message space S, so that in addition to (1) we have:

$$E_A(D_A(M))=M, \text{ for any message } M. \qquad (2)$$

(If the "cipher space" - the image of the message space S under E_A - is different from S then (1) need not imply (2), since D_A may not even be defined for those elements of the message space which are not in the cipher space.)

Suppose now that user A wants to send user B a "signed" document M. User A then sends $E_B(D_A(M))$ to B, who then deciphers it with D_B to obtain $M'=D_A(M)$. Now using E_A (available on the public file), B can read the "signed" document $E_A(M') = E_A(D_A(M)) = M$. Here M' will act as A's "signature" for the message M.

User A can not deny having sent B this message, since no one but A could have created $M'=D_A(M)$, under our assumption that D_A is not computable from E_A. User B can obviously convince a "judge" that $E_A(M')=M$, so that B has "proof" that A has signed the document.

Clearly B can not modify M to a different version M", since then B would have to create the corresponding signature $D_A(M")$ as well. Therefore B has received a document "signed" by A, which he can "prove" that A sent, but which B can not modify in any detail. (Nor can B forge A's signature on any other document).

We observe that the act of sending a "signed" message does not increase the length of the transmitted version of the message (compared to its "unsigned" form) at all, since the "signature" is effected by performing a length-preserving transformation on the message before transmission. A very long message should be broken into blocks, each block labelled with a "this is block i of n" notation, and transmitted after "signing" each block separately.

The concept of a public-key cryptosystem as described above, and its potential use as a means of implementing digital "signatures", are due to Diffie and Hellman[2]. The reader is encouraged to read this excellent article for further background and elaboration of this concept, as well as for a discussion of other problems in the area of cryptography. Their article was the motivation for the present work, in that while they presented the concept of a public-key cryptosystem, they did not present any practical way of implementing such a system. In this paper we present a candidate implementation scheme.

If the security of the system proposed here turns out to be satisfactory, then we will as a corollary have demonstrated the existence of "trap-door one-way functions", as defined in [2]. A "trap-door one-way function" is a function which is easy to compute and easy to invert, but for which the inverse function is difficult to compute from a description of the function itself.

II. Implementation

The scheme presented here enciphers a message M by raising it to a fixed power s modulo a certain composite number r. The deciphering operation is performed by raising the received message to another power t, again modulo r. User A makes public r and s, and keeps t private. (These values should more properly be denoted r_A, s_A, and t_A, since each user will have a separate set of values, but in what follows we will only concern outselves with user A's system, and will omit the subscripts.) We assume that the message can be viewed as a number less than r, or that it can be broken into a series of blocks, each of which can be viewed as a number less than r which will be separately enciphered.

We observe that raising a number x to the s-th power modulo r requires only $O(\log_2(r) \cdot T(r))$ operations to perform, if s is less than r, where $T(r)$ denotes the time required to multiply two numbers modulo r. This bound is easily derived by considering the binary representation of s, reading from left to right, as a rule for obtaining x^s from 1 by treating each 1 as an instruction to "square the preceding value and multiply the result by x", and each 0 as an instruction to "square the preceding value". Thus we may consider enciphering and deciphering to be "efficient" operations. The fact that the enciphering and deciphering operations are similar leads to a simple implementation (conceivably the whole operation could be implemented on a single integrated circuit chip).

As a small running example, consider the case

$$r = 47 \cdot 59 = 2773, \quad s = 17.$$

With an r of this size we can encode two english letters in a block, by mapping each letter into a two-digit number (blank=00, A=01, B=02, ..., Z=26). Thus the message

ITS ALL GREEK TO ME

(Julius Caesar, I,ii,288) would be encoded into ten blocks:

0920 1900 0112 1200 0718 0505 1100 2015 0013 0500 .

Since $s=10001$ in binary, the encoding of the first block goes as:

$$
\begin{aligned}
M^0 &= 1,\\
M^1 &= (1)^2 \cdot 920 = 920, \text{ (for the leftmost 1)}\\
M^2 &= (920)^2 = 635,\\
M^4 &= (635)^2 = 1140,\\
M^8 &= (1140)^2 = 1836,\\
M^{17} &= (1836)^2 \cdot 920 = 948.
\end{aligned}
$$

(Here all arithmetic is done modulo 2773.) In a similar fashion, the whole message is enciphered as:

0948 2342 1084 1444 2663 2390 0778 0774 0219 1655 .

In order to create a realistically-sized public-key cryptosystem, we use the fact that to determine whether a given integer n is prime or not can also be performed efficiently, even if n is over 100 digits long. As an illustration of the kind of test used by these procedures, the algorithms described in [4,8] are based on the following facts. For every prime number p and every number a not congruent to zero, mod p, we have

$$a^{p-1} \equiv 1 \pmod{p} \qquad (3)$$

On the other hand, for most composite numbers n at least one-half of the numbers a, $0<a<n$, fail to satisfy the analagous relation

$$a^{n-1} \equiv 1 \pmod{n}. \qquad (4)$$

Once an a which violates (4) is found we have "proof" that n is in fact composite. For example, since $2^{2772} \equiv 1088 \pmod{2773}$, we know that 2773 is composite. We refer the reader to the original papers discussing these results [4,6,8,9] for a detailed discussion of these procedures, including the appropriate tests to use for those numbers n which satisfy (4) for all a (the Carmichael numbers).

It is important to note that the efficient primality-testing algorithms just described do not in general, when given as input a composite integer n, determine any of the factors of n. It is somewhat surprising that while it is relatively easy to determine whether n is prime or composite, there are no efficient ways known for computing the prime factorization of a composite number n. To determine the factorization of an integer n which is the product of just two 50-digit prime numbers is considerably beyond current capabilities. Knuth[3, section 4.5.4] gives an excellent presentation of several efficient factoring algorithms. The most efficient general factoring algorithm known to the authors is due to Pollard [6]; it will factor a number n in $O(n^{1/4})$ steps. A 125-digit number can be tested for primality in about one minute; we estimate that factoring a number of that size could require 40 quadrillion years using Pollard's algorithm. If we may quote J. Brillhart, D. H. Lehmer, and J. L. Selfridge on the difficulty of factoring,

> "In general nothing but frustration can be expected to come from an attack on a number of 50 or more digits, even with the speeds available with modern computers." [1, page 645]

Let d be an integer such that determining the prime factorization of a number n which is the product of just two prime numbers of length d (in digits) is "computationally impossible". Choosing $d=40$ seems to be satisfactory at present. If better factoring algorithms are discovered then the appropriate value of d would have to be increased, but as long as testing for primality is significanly easier than factoring the scheme to be described will have the desired properties.

When user A desires to put on the public file his enciphering key, consisting of the integers r and s, he does so by determining two d-digit "random" prime numbers p and q, and an integer s which is relatively prime to $(p-1)\cdot(q-1)$. (The reason for this condition will be explained shortly.) Then A puts on public file the integers r and s, where r is defined to be $p\cdot q$. By assumption, only A will have available the prime factors p and q of r, even though r is on the public file. When A makes r and s public, the values of p and q are effectively hidden from everyone else due to the computational impossibility of factoring r in a reasonable amount of time.

For our example we have $p = 47$, $q = 59$, $r = p\cdot q = 2773$.

The subtask of finding a d-digit "random" prime number is easily accomplished by first generating an (odd) d-digit random number and then incrementing it by 2 until a prime number is found. By the prime number theorem, we should expect to have to do about $O(d)$ incrementations before finding a prime. In order to avoid those few cases where the efficient primality-testing algorithms <u>do</u> yield a factor, it is desirable to ensure that both $(p-1)$ and $(q-1)$ themselves contain large prime factors and that $\gcd(p-1,r-1)$ and $\gcd(q-1,r-1)$ are both small. The latter condition is easily checked. To obtain a prime number p such that $(p-1)$ has a large prime factor one can generate a d-digit prime number u and then find the first prime in the sequence $i\cdot u + 1$, for $i=2,4,6,\ldots$. By the prime number theorem for arithmetic progressions we can expect to find a prime after examining $O(d)$ elements of this series. (There is some additional security provided by selecting u in the same manner to be of the form $j \cdot v + 1$, where v is a large prime.)

The enciphering algorithm E_A is thus the operation:

$$E_A(M) \equiv M^s \pmod r$$

for any message M.

To obtain the corresponding deciphering algorithm, we will use the identity (due to Euler and Fermat) that for any message M which is relatively prime to r:

$$M^{\phi(r)} \equiv 1 \pmod r, \qquad (5)$$

where $\phi(r)$ is the Euler totient function giving the number of positive integers less than r which are relatively prime to r. Equation (5) is easily proved: the set of residues (mod r) which are relatively prime to r form a group of order $\phi(r)$ under multiplication, and in any group the order of an element must divide the order of the group. Since $\phi(p) = p-1$ for prime numbers p, equation (3) is a special case of (5). In our case, we have

$$\begin{aligned} \phi(r) &= \phi(p) \cdot \phi(q), \qquad (6) \\ &= (p-1) \cdot (q-1) \\ &= p \cdot q - (p+q) + 1 \end{aligned}$$

by the elementary properties of the totient function [5]. In our example we have

$$\phi(2773) = 46 \cdot 58 = 2668.$$

It is easy to see that the factorization of r enables the computation of $\phi(r)$ by (6), and that conversely the ability to compute $\phi(r)$ enables the factorization of r, since $(p+q)$ is easily obtained from r and $\phi(r)$, and $(p-q)$ can be obtained by taking the square root of $(p+q)^2 - 4pq$. By our assumptions about the size of d, therefore, it is not possible for anyone except A to know $\phi(r)$.

Since s is relatively prime with respect to $\phi(r)$, it has a multiplicative inverse t in the ring of integers modulo $\phi(r)$. Thus we have that

$$s \cdot t \equiv 1 \pmod{\phi(r)}.$$

The value of t is easily computed using a simple variant of Euclid's algorithm to compute the greatest common divisor of s and $\phi(r)$. (See exercise 4.5.2.15 in [3].) Briefly, the procedure is as follows. Euclid's algorithm computes $\gcd(x_0, x_1)$ by computing a series

$x_0, x_1, \ldots, x_k$ where $x_{i+1} = x_{i-1} \bmod x_i$ and $x_k = \gcd(x_0, x_1)$. It is simple to compute in addition for each x_i coefficients a_i and b_i such that $x_i = a_i \cdot x_0 + b_i \cdot x_1$. If $x_k = 1$ then b_k is the inverse of $x_1 \bmod x_0$. For our example we have

$$\begin{aligned}
&x_0 = 2668, \quad a_0 = 1, \quad b_0 = 0,\\
&x_1 = 17, \quad a_1 = 0, \quad b_1 = 1,\\
&x_2 = 16, \quad a_2 = 1, \quad b_2 = -156, \text{ (since } 2668 = 156 \cdot 17 + 16\text{)},\\
&x_3 = 1, \quad a_3 = -1, \quad b_3 = 157 \text{ (since } 17 = 1 \cdot 16 + 1\text{)}.
\end{aligned}$$

Therefore the inverse of 17(mod 2668) is 157.

It is now easy to see that

$$\begin{aligned}
(E_A(M))^t &\equiv (M^s)^t \pmod r\\
&\equiv M^{s \cdot t} \pmod r\\
&\equiv M^{u \cdot \phi(r)+1} \pmod r\\
&\equiv M^1 \equiv M \pmod r,
\end{aligned}$$

for some integer u. Therefore the deciphering function

$$D_A(C) \equiv C^t \pmod r$$

is the desired inverse operation. (The reader can check in our example that $948^{157} \equiv 920 \pmod{2773}$.)

It should of course be checked that t is large enough so that a direct search for it is infeasible. The value of s is rather arbitrary but should be chosen larger than $\log_2(r)$, so that every message suffers some "wrap-around" (reduction mod r) during the encoding process.

The preceding analysis was based on the assumption that the input message M was relatively prime to r. While not all numbers less than r are relatively prime to r, only those which are multiples of either p or q are not. Therefore the chances of finding, among a collection of messages, one which is not relatively prime to r is very small, say on the order of 10^{-d}, and is therefore negligible. This must be so by our assumption since if it were likely or easy to find a number less than r which was not relatively prime to r, then r could be factored. (The gcd of this number and r will be either p or q.)

It is interesting to note that the enciphering operation $E_A(M)$ is always invertible, even if the message M is a multiple of p (or similarly, q). The deciphering operation is modified as follows. We first note that if M is a multiple of p then so is $E_A(M)$. The decoder can detect this fact easily. If the decoder receives a multiple of p it concludes that M is a multiple of p, so that in order to determine M uniquely it need only determine the residue of M modulo q, by the Chinese remainder theorem. The residue of M modulo q can be found by:

$$M \equiv (E_A(M))^{t'} \pmod q,$$

where t' is the inverse of s modulo $(q-1)$. The existence of t' is guaranteed by the fact that s is relatively prime to $(p-1)\cdot(q-1)$, and therefore to $q-1$. To decoded a received message which is a multiple of p it therefore suffices to raise it to the t'-th power, modulo r. In our example the inverse of 17, mod 58, is 41. Thus the value $t'=41$ (instead of 157) would be used to decode those messages which are a multiple of $p=47$. Similarly those received messages which are a multiple of $q=59$ can be decoded with a value of $t'=19$.

III. Remarks

We note that a minor awkwardness exists in using our system for digital signatures in the fashion proposed by Diffie and Hellman. Namely, it may be necessary to "reblock" the signed message for encryption since the value of r used for signatures may be larger than that used for enciphering (every user has his own value of r). If desired, this problem can be avoided as follows. A certain threshold value h is decided upon (say $h = 10^{100}$). Every user then maintains two r,s pairs in the public file, one for enciphering purposes and one for signature purposes. If every user's signature r is less than h, and every user's enciphering r is greater than h, then reblocking in order to encipher a signed message will never be necessary.

We now examine this scheme from the viewpoint of the "enemy cryptanalyst" who wants to "break the system", that is, to find an efficient way of computing D_A given only r and s to work with. By our previous assumptions he can not do it in the same way that A did, since he does not have $\phi(r)$ available to him. He has two approaches he may try: (1) determine t or some equivalent number in some fashion that does not require the knowledge of $\phi(r)$, or (2) find an altogether different method of computing D_A.

A method for determining t is unlikely to exist since it would more or less enable a calculation of $\phi(r)$, since it is a factor of $s\cdot t - 1$. More precisely, a method for calculating a t corresponding to an arbitrary s would thus enable the cryptanalyst to determine many different multiples of $\phi(r)$, by varying s. The gcd of these quantities is likely to be $\phi(r)$. In any case Gary Miller [4] has in fact shown that determining any multiple of $\phi(r)$ enables r to be factored.

As for the the second approach, we have no proof that this is infeasible, nor is this a "well-known" computationally intractable problem. However, we feel reasonably confident that this is the case. Just as any modern cryptographic system must be "certified" by proving itself immune to a sophisticated cryptanalytic attack, the scheme proposed here must be similarly certified by having the preceding conjecture of intractability withstand a concerted attempt to disprove it. The reader is hereby challenged to find a way to "break" this scheme.

IV. Acknowledgements

We should like to thank Steve Boyack, Martin Hellman, and Abraham Lempel for several helpful discussions on these matters.

V. References

[1] Brillhart, J., D. H. Lehmer, and J. L. Selfridge. "New Primality Criteria and Factorizations of $2^m \pm 1$", Math. Comp. 29, 130(April 1975), 620-647.

[2] Diffie, W. and M. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory (Nov. 1976)

[3] Knuth, Donald E., "Seminumerical Algorithms" (Volume 2 of The Art of Computer Programming. Addison Wesley. Reading, Masssachusetts. 1969.)

[4] Miller, G. L. "Riemann's Hypothesis and Tests for Primality". Proceedings of the Seventh Annual ACM Symposium on the Theory of Computing. (Albuquerque, New Mexico, May 1975), 234-239. (An extended version of this paper is available as Research Report CS-75-27 from the Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada (Oct.,1975).)

[5] Niven, I., and H. S. Zuckerman. An Introduction to the Theory of Numbers. (John Wiley & Sons, New York 1972).

[6] Pollard, J.M. "Theorems on factorization and primality testing," Proc. Camb. Phil. Soc. (1974), 521-528.

[7] Potter, R. J., "Electronic Mail", Science 195,4283(18 March 1977), 1160-1164.

[8] Rabin, M. O., "Probabilistic Algorithms", in Algorithms and Complexity, edited by J. F. Traub (Academic Press, New York, 1976), 21-40.

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

Official Distribution List

Defense Documentation Center
Cameron Station
Alexandria, Va 22314 — 12 copies

Office of Naval Research
Information Systems Program
Code 437
Arlington, Va 22217 — 2 copies

Office of Naval Research
Code 102IP
Arlington, Va 22217 — 6 copies

Office of Naval Research
Code 200
Arlington, Va 22217 — 1 copy

Office of Naval Research
Code 455
Arlington, Va 22217 — 1 copy

Office of Naval Research
Code 458
Arlington, Va 22217 — 1 copy

Office of Naval Research
Branch Office, Boston
495 Summer Street
Boston, Ma 02210 — 1 copy

Office of Naval Research
Branch Office, Chicago
536 South Clark Street
Chicago, Il 60605 — 1 copy

Office of Naval Research
Branch Office, Pasadena
1030 East Green Street
Pasadena, Ca 91106 — 1 copy

New York Area Office
715 Broadway - 5th floor
New York, N. Y. 10003 — 1 copy

Naval Research Laboratory
Technical Information Division
Code 2627
Washington, D. C. 20375 — 6 copies

Dr. A. L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps
(Code RD-1)
Washington, D. C. 20380 — 1 copy

Naval Electronics Laboratory Center
Advanced Software Technology Division
Code 5200
San Diego, Ca 92152 — 1 copy

Mr. E. H. Gleissner
Naval Ship Research & Development Center
Computation & Mathematics Department
Bethesda, Md 20084 — 1 copy

Captain Grace M. Hopper
NAICOM/MIS Planning Branch (OP-916D)
Office of Chief of Naval Operations
Washington, D. C. 20350 — 1 copy

Mr. Kin B. Thompson
Technical Director
Information Systems Division (OP-91T)
Office of Chief of Naval Operations
Washington, D. C. 20350 — 1 copy