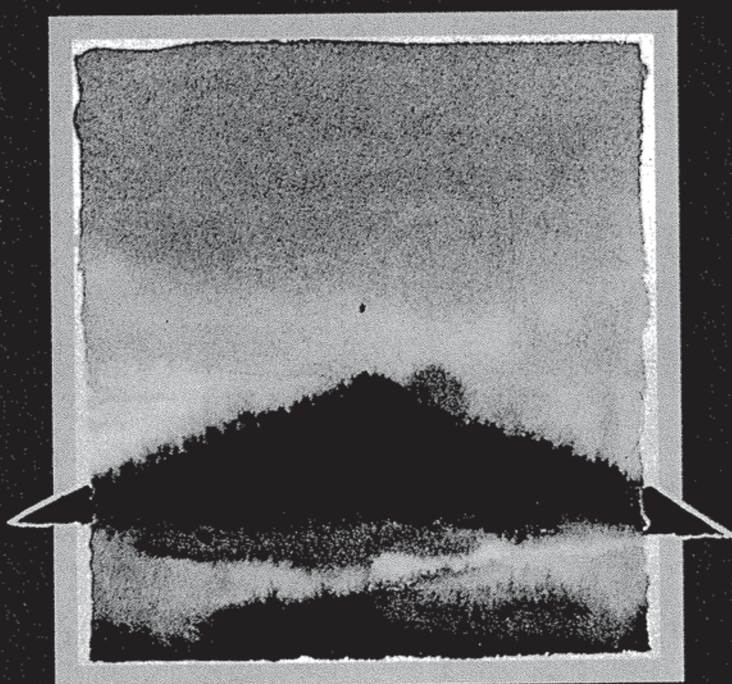
APPLIED CRYPTOGRAPHY



Protocols, Algorithms, and Source Code in C

BRUCE SCHNEIER

Associate Publisher: Katherine Schowalter
Editor: Paul Farrell
Managing Editor: Beth Austin
Editorial Production & Design: Editorial Services of New England, Inc.

This text is printed on acid-free paper.

Library of Congress Cataloging-in-Publication Data
Schneier, Bruce
Applied cryptography : protocols, algorithms, and source code in C / Bruce Schneier.
p. cm.
Includes bibliographical references and index.
ISBN 0-471-59756-2 (paper)
1. Computer security. 2. Telecommunication—security measures. 3. Cryptography. 4. Title

QA76.9.A25S35 1993 93-2139
005.8'2—dc20 CIP

Printed in the United States of America
10 9 8 7 6 5 4

makes it far less useful than the other algorithms discussed here. And even worse, considering the ease with which all the other variations fell, it doesn't seem prudent to trust them.

Patents

The original Merkle-Hellman algorithm is patented in the United States [428] and worldwide (see Table 12.1). PKP licenses the patent, along with other public-key cryptography patents. Anyone interested in obtaining a license should contact:

Robert B. Fougner
Director of Licensing
Public Key Partners
130 B Kifer Court
Sunnyvale, CA 94086
Tel: (408) 735-6779

The U.S. patent will expire on August 19, 1997.

TABLE 12.1
Foreign Merkle-Hellman Knapsack Patents

COUNTRY	NUMBER	DATE
Belgium	871039	5 Apr 1979
Netherlands	7810063	10 Apr 1979
Great Britain	2006580	2 May 1979
Germany	2843583	10 May 1979
Sweden	7810478	14 May 1979
France	2405532	8 Jun 1979
Germany	2843583	3 Jun 1982
Germany	2857905	15 Jul 1982
Canada	1128159	20 Jul 1982
Great Britain	2006580	18 Aug 1982
Switzerland	63416114	14 Jan 1983
Italy	1099780	28 Sep 1985

12.4 RSA

Soon after Merkle's knapsack algorithm came the first full-fledged public-key algorithm, one that works for encryption as well as for digital signatures. Of all the public-key algorithms proposed over the years, it is by far the easiest to understand and implement. (Martin Gardner published an early description of the algorithm in his "Mathematical Games" column in *Scientific American* [365].) It is also the most popular. Named after the three inventors, Ron Rivest, Adi Shamir, and Leonard Adleman, who first introduced the algorithm in 1978 [749,750], it has since withstood years of extensive cryptanalysis. Although the cryptanalysis neither proved nor disproved RSA's security, it does suggest a confidence level in the theoretical underpinnings of the algorithm.

RSA gets its security from the difficulty of factoring large numbers. The public and private keys are functions of a pair of large (100 to 200 digits or even larger) prime numbers. Recovering the plaintext from one of the keys and the ciphertext is conjectured to be equivalent to factoring the product of the two primes.

To generate the two keys, choose two large prime numbers, p and q. Compute the product:

$$n = p \times q$$

Then randomly choose the encryption key, e, such that e and $(p-1) \times (q-1)$ are relatively prime. Finally, use Euclid's algorithm to compute the decryption key, d, such that

$$e \times d = 1 \pmod{(p-1) \times (q-1)}$$

In other words,

$$d = e^{-1} \pmod{(p-1) \times (q-1)}$$

Note that d and n are also relatively prime. The numbers e and n are the public key; the number d is the private key. The two primes, p and q, are no longer needed. They should be discarded, but never revealed.

To encrypt a message m, first divide it into numerical blocks such that each block has a unique representation modulo n (with binary data, choose the largest power of 2 less than n). That is, if both p and q are 100-digit primes, then n will have just under 200 digits, and each message block, m_i, should be just under 200 digits long. The encrypted message, c, will be made up of similarly sized message blocks, c_i, of about the same length. The encryption formula is simply:

$$c_i = m_i^e \pmod{n}$$

To decrypt a message, take each encrypted block c_i and compute:

$$m_i = c_i^d \pmod{n}$$

Since:

$$c_i^d = (m_i^e)^d = m_i^{ed} = m_i^{k(p-1)(q-1)+1} = m_i \times m_i^{k(p-1)(q-1)} = m_i \times 1 = m_i,$$
$$\text{all mod } n$$

the formula recovers the message. This is summarized in Table 12.2.

TABLE 12.2
RSA Encryption

PUBLIC KEY:	
n	product of two primes, p and q (p and q must remain secret)
e	relatively prime to $(p-1)\times(q-1)$
PRIVATE KEY:	
d	e^{-1} (mod $(p-1)\times(q-1)$)
ENCRYPTING:	
	$c = m^e$ (mod n)
DECRYPTING:	
	$m = c^d$ (mod n)

The message could just as easily have been encrypted with d and decrypted with e; the choice is arbitrary. I am not including the number theory that proves why this works; most current texts on cryptography cover the theory in detail.

A short example will probably go a long way to making this clearer. If $p = 47$ and $q = 71$, then

$$n = p \times q = 3337$$

The encryption key e must have no factors in common with:

$$(p-1)\times(q-1) = 46 \times 70 = 3220$$

Choose e (at random) to be 79. In that case:

$$d = 79^{-1} \text{ (mod 3220)} = 1019$$

This number was calculated using the extended Euclidean algorithm (see Section 9.3). Publish e and n, and keep d secret. Discard p and q.

To encrypt the message

$$m = 6882326879666683$$

first break it into small blocks. Three-digit blocks work nicely in this case. The message will be encrypted in six blocks, m_i, in which:

$$m_1 = 688$$
$$m_2 = 232$$
$$m_3 = 687$$
$$m_4 = 966$$
$$m_5 = 668$$
$$m_6 = 3$$

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase®
Smarter legal research.