

Keyword

☒ Brief ☐ Full

[Advanced Search](#)

[Search Tips](#)

 **BROWSE**
BY TOPIC

To access the contents, click the chapter and section titles.

Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)

(Publisher: John Wiley & Sons, Inc.)

Author(s): Bruce Schneier

ISBN: 0471128457

Publication Date: 01/01/96

Bookmark It

Search this book:

[Foreword by Whitfield Diffie](#)

[Preface](#)

[About the Author](#)

[Chapter 1—Foundations](#)

[1.1 Terminology](#)

[1.2 Steganography](#)

[1.3 Substitution Ciphers and Transposition Ciphers](#)

[1.4 Simple XOR](#)

[1.5 One-Time Pads](#)

[1.6 Computer Algorithms](#)

[1.7 Large Numbers](#)

[Part I—Cryptographic Protocols](#)

[Chapter 2—Protocol Building Blocks](#)

[2.1 Introduction to Protocols](#)

[2.2 Communications Using Symmetric Cryptography](#)

[2.3 One-Way Functions](#)

[2.4 One-Way Hash Functions](#)

[2.5 Communications Using Public-Key Cryptography](#)

[2.6 Digital Signatures](#)

[2.7 Digital Signatures with Encryption](#)

[2.8 Random and Pseudo-Random-Sequence Generation](#)

Chapter 3—Basic Protocols

[3.1 Key Exchange](#)

[3.2 Authentication](#)

[3.3 Authentication and Key Exchange](#)

[3.4 Formal Analysis of Authentication and Key-Exchange Protocols](#)

[3.5 Multiple-Key Public-Key Cryptography](#)

[3.6 Secret Splitting](#)

[3.7 Secret Sharing](#)

[3.8 Cryptographic Protection of Databases](#)

Chapter 4—Intermediate Protocols

[4.1 Timestamping Services](#)

[4.2 Subliminal Channel](#)

[4.3 Undeniable Digital Signatures](#)

[4.4 Designated Confirmer Signatures](#)

[4.5 Proxy Signatures](#)

[4.6 Group Signatures](#)

[4.7 Fail-Stop Digital Signatures](#)

[4.8 Computing with Encrypted Data](#)

[4.9 Bit Commitment](#)

[4.10 Fair Coin Flips](#)

[4.11 Mental Poker](#)

[4.12 One-Way Accumulators](#)

[4.13 All-or-Nothing Disclosure of Secrets](#)

[4.14 Key Escrow](#)

Chapter 5—Advanced Protocols

[5.1 Zero-Knowledge Proofs](#)

[5.2 Zero-Knowledge Proofs of Identity](#)

[5.3 Blind Signatures](#)

[5.4 Identity-Based Public-Key Cryptography](#)

[5.5 Oblivious Transfer](#)

[5.6 Oblivious Signatures](#)

[5.7 Simultaneous Contract Signing](#)

[5.8 Digital Certified Mail](#)

[5.9 Simultaneous Exchange of Secrets](#)

Chapter 6—Esoteric Protocols

[6.1 Secure Elections](#)

[6.2 Secure Multiparty Computation](#)

[6.3 Anonymous Message Broadcast](#)

[6.4 Digital Cash](#)

Part II—Cryptographic Techniques

Chapter 7—Key Length

[7.1 Symmetric Key Length](#)

[7.2 Public-Key Key Length](#)

[7.3 Comparing Symmetric and Public-Key Key Length](#)

[7.4 Birthday Attacks against One-Way Hash Functions](#)

[7.5 How Long Should a Key Be?](#)

[7.6 Caveat Emptor](#)

Chapter 8—Key Management

[8.1 Generating Keys](#)

[8.2 Nonlinear Keyspaces](#)

[8.3 Transferring Keys](#)

[8.4 Verifying Keys](#)

[8.5 Using Keys](#)

[8.6 Updating Keys](#)

[8.7 Storing Keys](#)

[8.8 Backup Keys](#)

[8.9 Compromised Keys](#)

[8.10 Lifetime of Keys](#)

[8.11 Destroying Keys](#)

[8.12 Public-Key Key Management](#)

Chapter 9—Algorithm Types and Modes

[9.1 Electronic Codebook Mode](#)

[9.2 Block Replay](#)

[9.3 Cipher Block Chaining Mode](#)

[9.4 Stream Ciphers](#)

[9.5 Self-Synchronizing Stream Ciphers](#)

[9.6 Cipher-Feedback Mode](#)

[9.7 Synchronous Stream Ciphers](#)

[9.8 Output-Feedback Mode](#)

[9.9 Counter Mode](#)

[9.10 Other Block-Cipher Modes](#)

[9.11 Choosing a Cipher Mode](#)

[9.12 Interleaving](#)

[9.13 Block Ciphers versus Stream Ciphers](#)

[Chapter 10—Using Algorithms](#)

[10.1 Choosing an Algorithm](#)

[10.2 Public-Key Cryptography versus Symmetric Cryptography](#)

[10.3 Encrypting Communications Channels](#)

[10.4 Encrypting Data for Storage](#)

[10.5 Hardware Encryption versus Software Encryption](#)

[10.6 Compression, Encoding, and Encryption](#)

[10.7 Detecting Encryption](#)

[10.8 Hiding Ciphertext in Ciphertext](#)

[10.9 Destroying Information](#)

[Part III—Cryptographic Algorithms](#)

[Chapter 11—Mathematical Background](#)

[11.1 Information Theory](#)

[11.2 Complexity Theory](#)

[11.3 Number Theory](#)

[11.4 Factoring](#)

[11.5 Prime Number Generation](#)

[11.6 Discrete Logarithms in a Finite Field](#)

[Chapter 12—Data Encryption Standard \(DES\)](#)

[12.1 Background](#)

[12.2 Description of DES](#)

[12.3 Security of DES](#)

[12.4 Differential and Linear Cryptanalysis](#)

[12.5 The Real Design Criteria](#)

[12.6 DES Variants](#)

[12.7 How Secure Is DES Today?](#)

Chapter 13—Other Block Ciphers

[13.1 Lucifer](#)

[13.2 Madryga](#)

[13.3 NewDES](#)

[13.4 FEAL](#)

[13.5 REDOC](#)

[13.6 LOKI](#)

[13.7 Khufu and Khafre](#)

[13.8 RC2](#)

[13.9 IDEA](#)

[13.10 MMB](#)

[13.11 CA-1.1](#)

[13.12 Skipjack](#)

Chapter 14—Still Other Block Ciphers

[14.1 GOST](#)

[14.2 CAST](#)

[14.3 Blowfish](#)

[14.4 SAFER](#)

[14.5 3-Way](#)

[14.6 Crab](#)

[14.7 SXAL8/MBAL](#)

[14.8 RC5](#)

[14.9 Other Block Algorithms](#)

[14.10 Theory of Block Cipher Design](#)

[14.11 Using one-Way Hash Functions](#)

[14.12 Choosing a Block Algorithm](#)

Chapter 15—Combining Block Ciphers

[15.1 Double Encryption](#)

[15.2 Triple Encryption](#)

[15.3 Doubling the Block Length](#)

[15.4 Other Multiple Encryption Schemes](#)

[15.5 CDMF Key Shortening](#)

[15.6 Whitening](#)

[15.7 Cascading Multiple Block Algorithms](#)

[15.8 Combining Multiple Block Algorithms](#)

Chapter 16—Pseudo-Random-Sequence Generators and Stream Ciphers

[16.1 Linear Congruential Generators](#)

[16.2 Linear Feedback Shift Registers](#)

[16.3 Design and Analysis of Stream Ciphers](#)

[16.4 Stream Ciphers Using LFSRs](#)

[16.5 A5](#)

[16.6 Hughes XPD/KPD](#)

[16.7 Nanoteg](#)

[16.8 Rambutan](#)

[16.9 Additive Generators](#)

[16.10 Gifford](#)

[16.11 Algorithm M](#)

[16.12 PKZIP](#)

Chapter 17—Other Stream Ciphers and Real Random-Sequence Generators

[17.1 RC4](#)

[17.2 SEAL](#)

[17.3 WAKE](#)

[17.4 Feedback with Carry Shift Registers](#)

[17.5 Stream Ciphers Using FCSRs](#)

[17.6 Nonlinear-Feedback Shift Registers](#)

[17.7 Other Stream Ciphers](#)

[17.8 System-Theoretic Approach to Stream-Cipher Design](#)

[17.9 Complexity-Theoretic Approach to Stream-Cipher Design](#)

[17.10 Other Approaches to Stream-Cipher Design](#)

[17.11 Cascading Multiple Stream Ciphers](#)

[17.12 Choosing a Stream Cipher](#)

[17.13 Generating Multiple Streams from a Single Pseudo-Random-Sequence Generator](#)

[17.14 Real Random-Sequence Generators](#)

Chapter 18—One-Way Hash Functions

18.1 Background

18.2 Snefru

18.3 N- Hash

18.4 MD4

18.5 MD5

18.6 MD2

18.7 Secure Hash Algorithm (SHA)

18.8 RIPE-MD

18.9 HAVAL

18.10 Other One-Way Hash Functions

18.11 One-Way Hash Functions Using Symmetric Block Algorithms

18.12 Using Public-Key Algorithms

18.13 Choosing a One-Way Hash Function

18.14 Message Authentication Codes

Chapter 19—Public-Key Algorithms

19.1 Background

19.2 Knapsack Algorithms

19.3 RSA

19.4 Pohlig-Hellman

19.5 Rabin

19.6 ElGamal

19.7 McEliece

19.8 Elliptic Curve Cryptosystems

19.9 LUC

19.10 Finite Automaton Public-Key Cryptosystems

Chapter 20—Public-Key Digital Signature Algorithms

20.1 Digital Signature Algorithm (DSA)

20.2 DSA Variants

20.3 Gost Digital Signature Algorithm

20.4 Discrete Logarithm Signature Schemes

20.5 Ong-Schnorr-Shamir

20.6 ESIGN

[20.7 Cellular Automata](#)

[20.8 Other Public-Key Algorithms](#)

Chapter 21—Identification Schemes

[21.1 Feige-Fiat-Shamir](#)

[21.2 Guillou-Quisquater](#)

[21.3 Schnorr](#)

[21.4 Converting Identification Schemes to Signature Schemes](#)

Chapter 22—Key-Exchange Algorithms

[22.1 Diffie-Hellman](#)

[22.2 Station-to-Station Protocol](#)

[22.3 Shamir's Three-Pass Protocol](#)

[22.4 COMSET](#)

[22.5 Encrypted Key Exchange](#)

[22.6 Fortified Key Negotiation](#)

[22.7 Conference Key Distribution and Secret Broadcasting](#)

Chapter 23—Special Algorithms for Protocols

[23.1 Multiple-Key Public-Key Cryptography](#)

[23.2 Secret-Sharing Algorithms](#)

[23.3 Subliminal Channel](#)

[23.4 Undeniable Digital Signatures](#)

[23.5 Designated Confirmer Signatures](#)

[23.6 Computing with Encrypted Data](#)

[23.7 Fair Coin Flips](#)

[23.8 One-Way Accumulators](#)

[23.9 All-or-Nothing Disclosure of Secrets](#)

[23.10 Fair and Failsafe Cryptosystems](#)

[23.11 Zero-Knowledge Proofs of Knowledge](#)

[23.12 Blind Signatures](#)

[23.13 Oblivious Transfer](#)

[23.14 Secure Multiparty Computation](#)

[23.15 Probabilistic Encryption](#)

[23.16 Quantum Cryptography](#)

Part IV—The Real World

Chapter 24—Example Implementations

24.1 IBM Secret-Key Management Protocol

24.2 MITRENET

24.3 ISDN

24.4 STU-III

24.5 Kerberos

24.6 KryptoKnight

24.7 SESAME

24.8 IBM Common Cryptographic Architecture

24.9 ISO Authentication Framework

24.10 Privacy-Enhanced Mail (PEM)

24.11 Message Security Protocol (MSP)

24.12 Pretty Good Privacy (PGP)

24.13 Smart Cards

24.14 Public-Key Cryptography Standards (PKCS)

24.15 Universal Electronic Payment System (UEPS)

24.16 Clipper

24.17 Capstone

24.18 AT&T Model 3600 Telephone Security Device (TSD)

Chapter 25—Politics

25.1 National Security Agency (NSA)

25.2 National Computer Security Center (NCSC)

25.3 National Institute of Standards and Technology (NIST)

25.4 RSA Data Security, Inc.

25.5 Public Key Partners

25.6 International Association for Cryptologic Research (IACR)

25.7 RACE Integrity Primitives Evaluation (RIPE)

25.8 Conditional Access for Europe (CAFE)

25.9 ISO/IEC 9979

25.10 Professional, Civil Liberties, and Industry Groups

25.11 Sci.crypt

25.12 Cypherpunks

[25.13 Patents](#)

[25.14 U.S. Export Rules](#)

[25.15 Foreign Import and Export of Cryptography](#)

[25.16 Legal Issues](#)

[Afterword by Matt Blaze](#)

[Part V—Source Code](#)

[References](#)

[Index](#)

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)

All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

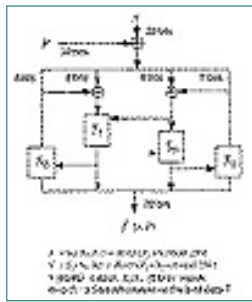


Figure 18.4 Function f .

The same attack can find pairs of messages that hash to the same value for 12-round N -Hash in 2^{56} operations, compared to 2^{64} operations for a brute-force attack. N -hash with 15 rounds is safe from differential cryptanalysis: The attack requires 2^{72} operations.

The algorithm's designers recommend using N -Hash with at least 8 rounds [1106]. Given the proven insecurity of N -Hash and FEAL (and its speed with 8 rounds), I recommend using another algorithm entirely.

18.4 MD4

MD4 is a one-way hash function designed by Ron Rivest [1318, 1319, 1321]. MD stands for **Message Digest**; the algorithm produces a 128-bit hash, or message digest, of the input message.

In [1319], Rivest outlined his design goals for the algorithm:

Security. It is computationally infeasible to find two messages that hashed to the same value. No attack is more efficient than brute force.

Direct Security. MD4's security is not based on any assumption, like the difficulty of factoring.

Speed. MD4 is suitable for high-speed software implementations. It is based on a simple set of bit manipulations on 32-bit operands.

Simplicity and Compactness. MD4 is as simple as possible, without large data structures or a complicated program.

Favor Little-Endian Architectures. MD4 is optimized for microprocessor architectures (specifically Intel microprocessors); larger and faster computers make any necessary translations.

After the algorithm was first introduced, Bert den Boer and Antoon Bosselaers successfully cryptanalyzed the last two of the algorithm's three rounds [202]. In an unrelated cryptanalytic result, Ralph Merkle successfully attacked the first two rounds [202]. Eli Biham discussed a differential cryptanalysis attack against the first two rounds of MD4 [159]. Even though these attacks could not be extended to the full algorithm, Rivest strengthened the algorithm. The result is MD5.

18.5 MD5

MD5 is an improved version of MD4 [1386, 1322]. Although more complex than MD4, it is similar in design and also produces a 128-bit hash.

Description of MD5

After some initial processing, MD5 processes the input text in 512-bit blocks, divided into 16 32-bit sub-blocks. The output of the algorithm is a set of four 32-bit blocks, which concatenate to form a single 128-bit hash value.

First, the message is padded so that its length is just 64 bits short of being a multiple of 512. This padding is a single 1-bit added to the end of the message, followed by as many zeros as are required. Then, a 64-bit representation of the message's length (before padding bits were added) is appended to the result. These two steps serve to make the message length an exact multiple of 512 bits in length (required for the rest of the algorithm), while ensuring that different messages will not look the same after padding.

Four 32-bit variables are initialized:

$A = 0x01234567$

$B = 0x89abcdef$

$C = 0xfedcba98$

$D = 0x76543210$

These are called **chaining variables**.

Now, the main loop of the algorithm begins. This loop continues for as many 512-bit blocks as are in the message.

The four variables are copied into different variables: a gets A , b gets B , c gets C , and d gets D .

The main loop has four rounds (MD4 had only three rounds), all very similar. Each round uses a different operation 16 times. Each operation performs a nonlinear function on three of a , b , c , and d . Then it adds that result to the fourth variable, a sub-block of the text and a constant. Then it rotates that result to the right a variable number of bits and adds the result to one of a , b , c , or d . Finally the result replaces one of a , b , c , or d . See Figures 18.5 and 18.6.

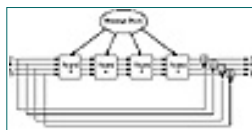


Figure 18.5 MD5 main loop.

There are four nonlinear functions, one used in each operation (a different one for each round).

$$F(X,Y,Z) = (X \text{ XOR } Y) \text{ OR } ((\neg X) \text{ AND } Z)$$

$$G(X,Y,Z) = (X \text{ AND } Z) \text{ OR } (Y \text{ AND } (\neg Z))$$

$$H(X,Y,Z) = X \text{ AND } Y \text{ AND } Z$$

$$I(X,Y,Z) = Y \text{ AND } (X \text{ OR } (\neg Z))$$

($\bullet$ is XOR, AND is AND, OR is OR, and $\neg$ is NOT.)

These functions are designed so that if the corresponding bits of X , Y , and Z are independent and unbiased, then each bit of the result will also be independent and unbiased. The function F is the bit-wise conditional: If X then

Y else Z. The function H is the bit-wise parity operator.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)

All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.

☒ Brief
 ☐ Full

[Advanced Search](#)

[Search Tips](#)

To access the contents, click the chapter and section titles.

Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)

(Publisher: John Wiley & Sons, Inc.)

Author(s): Bruce Schneier

ISBN: 0471128457

Publication Date: 01/01/96

Bookmark It

Search this book:

[Previous](#)
[Table of Contents](#)
[Next](#)

If M_j represents the j th sub-block of the message (from 0 to 15), and $\lll s$ represents a left circular shift of s bits, the four operations are:

$FF(a, b, c, d, M_j, s, t_i)$ denotes $a = b + ((a + F(b, c, d) + M_j + t_i) \lll s)$

$GG(a, b, c, d, M_j, s, t_i)$ denotes $a = b + ((a + G(b, c, d) + M_j + t_i) \lll s)$

$HH(a, b, c, d, M_j, s, t_i)$ denotes $a = b + ((a + H(b, c, d) + M_j + t_i) \lll s)$

$II(a, b, c, d, M_j, s, t_i)$ denotes $a = b + ((a + I(b, c, d) + M_j + t_i) \lll s)$

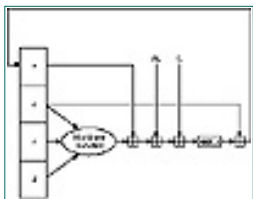


Figure 18.6 One MD5 operation.

The four rounds (64 steps) look like:

Round 1:

$FF(a, b, c, d, M_0, 7, 0xd76aa478)$

$FF(d, a, b, c, M_1, 12, 0xe8c7b756)$

$FF(c, d, a, b, M_2, 17, 0x242070db)$

$FF(b, c, d, a, M_3, 22, 0xc1bdcee)$

$FF(a, b, c, d, M_4, 7, 0xf57c0faf)$

$FF(d, a, b, c, M_5, 12, 0x4787c62a)$

$FF(c, d, a, b, M_6, 17, 0xa8304613)$

$FF(b, c, d, a, M_7, 22, 0xfd469501)$

$FF(a, b, c, d, M_8, 7, 0x698098d8)$

FF ($d, a, b, c, M_9, 12, 0x8b44f7af$)
FF ($c, d, a, b, M_{10}, 17, 0xffff5bb1$)
FF ($b, c, d, a, M_{11}, 22, 0x895cd7be$)
FF ($a, b, c, d, M_{12}, 7, 0x6b901122$)
FF ($d, a, b, c, M_{13}, 12, 0xfd987193$)
FF ($c, d, a, b, M_{14}, 17, 0xa679438e$)
FF ($b, c, d, a, M_{15}, 22, 0x49b40821$)

Round 2:

GG ($a, b, c, d, M_1, 5, 0xf61e2562$)
GG ($d, a, b, c, M_6, 9, 0xc040b340$)
GG ($c, d, a, b, M_{11}, 14, 0x265e5a51$)
GG ($b, c, d, a, M_0, 20, 0xe9b6c7aa$)
GG ($a, b, c, d, M_5, 5, 0xd62f105d$)
GG ($d, a, b, c, M_{10}, 9, 0x02441453$)
GG ($c, d, a, b, M_{15}, 14, 0xd8a1e681$)
GG ($b, c, d, a, M_4, 20, 0xe7d3fbc8$)
GG ($a, b, c, d, M_9, 5, 0x21e1cde6$)
GG ($d, a, b, c, M_{14}, 9, 0xc33707d6$)
GG ($c, d, a, b, M_3, 14, 0xf4d50d87$)
GG ($b, c, d, a, M_8, 20, 0x455a14ed$)
GG ($a, b, c, d, M_{13}, 5, 0xa9e3e905$)
GG ($d, a, b, c, M_2, 9, 0xfcefa3f8$)
GG ($c, d, a, b, M_7, 14, 0x676f02d9$)
GG ($b, c, d, a, M_{12}, 20, 0x8d2a4c8a$)

Round 3:

HH ($a, b, c, d, M_5, 4, 0xfffa3942$)
HH ($d, a, b, c, M_8, 11, 0x8771f681$)
HH ($c, d, a, b, M_{11}, 16, 0x6d9d6122$)
HH ($b, c, d, a, M_{14}, 23, 0xfde5380c$)
HH ($a, b, c, d, M_1, 4, 0xa4beea44$)
HH ($d, a, b, c, M_4, 11, 0x4bdecfa9$)
HH ($c, d, a, b, M_7, 16, 0xf6bb4b60$)
HH ($b, c, d, a, M_{10}, 23, 0xbebfb7c0$)
HH ($a, b, c, d, M_{13}, 4, 0x289b7ec6$)
HH ($d, a, b, c, M_0, 11, 0xeea127fa$)

HH ($c, d, a, b, M_3, 16, 0xd4ef3085$)

HH ($b, c, d, a, M_6, 23, 0x04881d05$)

HH ($a, b, c, d, M_9, 4, 0xd9d4d039$)

HH ($d, a, b, c, M_{12}, 11, 0xe6db99e5$)

HH ($c, d, a, b, M_{15}, 16, 0x1fa27cf8$)

HH ($b, c, d, a, M_2, 23, 0xc4ac5665$)

Round 4:

II ($a, b, c, d, M_0, 6, 0xf4292244$)

II ($d, a, b, c, M_7, 10, 0x432aff97$)

II ($c, d, a, b, M_{14}, 15, 0xab9423a7$)

II ($b, c, d, a, M_5, 21, 0xfc93a039$)

II ($a, b, c, d, M_{12}, 6, 0x655b59c3$)

II ($d, a, b, c, M_3, 10, 0x8f0ccc92$)

II ($c, d, a, b, M_{10}, 15, 0xffeff47d$)

II ($b, c, d, a, M_1, 21, 0x85845dd1$)

II ($a, b, c, d, M_8, 6, 0x6fa87e4f$)

II ($d, a, b, c, M_{15}, 10, 0xfe2ce6e0$)

II ($c, d, a, b, M_6, 15, 0xa3014314$)

II ($b, c, d, a, M_{13}, 21, 0x4e0811a1$)

II ($a, b, c, d, M_4, 6, 0xf7537e82$)

II ($d, a, b, c, M_{11}, 10, 0xbd3af235$)

II ($c, d, a, b, M_2, 15, 0x2ad7d2bb$)

II ($b, c, d, a, M_9, 21, 0xeb86d391$)

Those constants, t_i , were chosen as follows:

In step i , t_i is the integer part of $2^{32} \cdot \text{abs}(\sin(i))$, where i is in radians.

After all of this, a, b, c , and d are added to A, B, C, D , respectively, and the algorithm continues with the next block of data. The final output is the concatenation of A, B, C , and D .

Security of MD5

Ron Rivest outlined the improvements of MD5 over MD4 [1322]:

1. A fourth round has been added.
2. Each step now has a unique additive constant.
3. The function G in round 2 was changed from $((X \oplus Y) \mid (X \oplus Z) \mid (Y \oplus Z))$ to $((X \oplus Z) \mid (Y \oplus \neg Z))$ to make G less symmetric.
4. Each step now adds in the result of the previous step. This promotes a faster avalanche effect.

5. The order in which message sub-blocks are accessed in rounds 2 and 3 is changed, to make these patterns less alike.
6. The left circular shift amounts in each round have been approximately optimized, to yield a faster avalanche effect. The four shifts used in each round are different from the ones used in other rounds.

Tom Berson attempted to use differential cryptanalysis against a single round of MD5 [144], but his attack is ineffective against all four rounds. A more successful attack by den Boer and Bosselaers produces collisions using the compression function in MD5 [203, 1331, 1336]. This does not lend itself to attacks against MD5 in practical applications, and it does not affect the use of MD5 in Luby-Rackoff-like encryption algorithms (see Section 14.11). It does mean that one of the basic design principles of MD5—to design a collision-resistant compression function—has been violated. Although it is true that “there seems to be a weakness in the compression function, but it has no practical impact on the security of the hash function” [1336], I am wary of using MD5.

18.6 MD2

MD2 is another 128-bit one-way hash function designed by Ron Rivest [801, 1335]. It, along with MD5, is used in the PEM protocols (see Section 24.10). The security of MD2 is dependent on a random permutation of bytes. This permutation is fixed, and depends on the digits of π . $S_0, S_1, S_2, \dots, S_{255}$ is the permutation. To hash a message M :

- (1) Pad the message with i bytes of value i so that the resulting message is a multiple of 16 bytes long.
- (2) Append a 16-byte checksum to the message.
- (3) Initialize a 48-byte block: $X_0, X_1, X_2, \dots, X_{47}$. Set the first 16 bytes of X to be 0, the second 16 bytes of X to be the first 16 bytes of the message, and the third 16 bytes of X to be the XOR of the first 16 bytes of X and the second 16 bytes of X .
- (4) This is the compression function:

$t = 0$
 For $j = 0$ to 17
 For $k = 0$ to 47
 $t = X_k \text{ XOR } S_t$
 $X_k = t$
 $t = (t + j) \bmod 256$
- (5) Set the second 16 bytes of X to be the second 16 bytes of the message, and the third 16 bytes of X to be the XOR of the first 16 bytes of X and the second 16 bytes of X . Do step (4). Repeat steps (5) and (4) with every 16 bytes of the message, in turn.
- (6) The output is the first 16 bytes of X .

Although no weaknesses in MD2 have been found (see [1262]), it is slower

than most other suggested hash functions.

Previous	Table of Contents	Next
--------------------------	-----------------------------------	----------------------

[Products](#) | [Contact Us](#) | [About Us](#) | [Privacy](#) | [Ad Info](#) | [Home](#)

Use of this site is subject to certain [Terms & Conditions](#), [Copyright © 1996-2000 EarthWeb Inc.](#)

All rights reserved. Reproduction whole or in part in any form or medium without express written [permission](#) of EarthWeb is prohibited. Read EarthWeb's [privacy](#) statement.