FIPS PUB 180-1 - FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION

(Supersedes FIPS PUB 180 - 1993 May 11)

1995 April 17

U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology

SECURE HASH STANDARD

/*** NOTE: NOT OFFICIAL. HARD COPY IS THE OFFICIAL VERSION.

^ is used for exponentiation or superscript. ***/

CATEGORY: COMPUTER SECURITY

U.S. DEPARTMENT OF COMMERCE, Ronald H. Brown, Secretary

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

Foreword

The Federal Information Processing Standards Publication Series of the National Institute of Standards and Technology (NIST) is the official series of publications relating to standards and guidelines adopted and promulgated under the provisions of Section 111(d) of the Federal Property and Administrative Services Act of 1949 as amended by the Computer Security Act of 1987, Public Law 100-235. These mandates have given the Secretary of Commerce and NIST important responsibilities for improving the utilization and management of computer and related telecommunications systems in the Federal Government. The NIST, through the Computer Systems Laboratory, provides leadership, technical guidance, and coordination of Government efforts in the development of standards and guidelines in these areas.

Comments concerning Federal Information Processing Standards Publications are welcomed and should be addressed to the Director, Computer Systems Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899.

James H. Burrows, Director

Computer Systems Laboratory

Abstract

This standard specifies a Secure Hash Algorithm (SHA-1) which can be used to generate a condensed representation of a message called a message digest. The SHA-1 is required for use with the Digital Signature Algorithm (DSA) as specified in the Digital Signature Standard (DSS) and whenever a secure hash algorithm is required for Federal applications. The SHA-1 is used by both the transmitter and intended receiver of a message in computing and verifying a digital signature.

Key words: computer security; digital signatures; Federal Information Processing Standard (FIPS); hash algorithm.

FIPS PUB 180-1

Federal Information

Processing Standards Publication 180-1

1995 APRIL 17

ANNOUNCING THE

SECURE HASH STANDARD

Federal Information Processing Standards Publications (FIPS PUBS) are issued by the National Institute of Standards and Technology (NIST) after approval by the Secretary of Commerce pursuant to Section 111(d) of the Federal Property and Administrative Services Act of 1949 as amended by the Computer Security Act of 1987, Public Law 100-235.

Name of Standard: Secure Hash Standard.

Category of Standard: Computer Security.

Explanation: This Standard specifies a Secure Hash Algorithm, SHA-1, for computing a condensed representation of a message or a data file. When a message of any length < 2^64 bits is input, the SHA-1 produces a 160-bit output called a message digest. The message digest can then be input to the Digital Signature Algorithm (DSA) which generates or verifies the signature for the message. Signing the message digest rather than the message often improves the efficiency of the process because the message digest is usually much smaller in size than the message. The same hash algorithm must be used by the verifier of a digital signature as was used by the creator of the digital signature.

The SHA-1 is called secure because it is computationally infeasible to find a message which corresponds to a given message digest, or to find two different messages which produce the same message digest. Any change to a message in transit will, with very high probability, result in a different message digest, and the signature will fail to verify. SHA-1 is a technical revision of SHA (FIPS 180). A circular left shift operation has been added to the specifications in section 7, line b, page 9 of FIPS 180 and its equivalent in section 8, line c, page 10 of FIPS 180. This revision improves the security provided by this standard. The SHA-1 is based on principles similar to those used by Professor Ronald L. Rivest of MIT when designing the MD4 message digest algorithm ("The MD4 Message Digest Algorithm," Advances in Cryptology - CRYPTO '90 Proceedings, Springer-Verlag, 1991, pp. 303-311), and is closely modelled after that algorithm.

Approving Authority: Secretary of Commerce.

Maintenance Agency: U.S. Department of Commerce, National Institute of Standards and Technology, Computer Systems Laboratory.

Applicability: This standard is applicable to all Federal departments and agencies for the protection of unclassified information that is not subject to section 2315 of Title 10, United States Code, or section 3502(2) of Title 44, United States Code. This standard is required for use with the Digital Signature Algorithm (DSA) as specified in the Digital Signature Standard (DSS) and whenever a secure hash algorithm is required for Federal applica- tions. Private and commercial organizations are encouraged to adopt and use this standard.

Applications: The SHA-1 may be used with the DSA in electronic mail, electronic funds transfer, software distribution, data storage, and other applications which require data integrity assurance and data origin authentication. The SHA-1 may also be used whenever it is necessary to generate a condensed version of a message.

Implementations: The SHA-1 may be implemented in software, firmware, hardware, or any combination thereof. Only implementations of the SHA-1 that are validated by NIST will be considered as complying with this standard. Information about the requirements for validating implementations of this standard can be obtained from the National Institute of Standards and Technology, Computer Systems Laboratory, Attn: SHS Validation, Gaithersburg, MD 20899.

Export Control: Implementations of this standard are subject to Federal Government export controls as specified in Title 15, Code of Federal Regulations, Parts 768 through 799. Exporters are advised to contact the Department of Commerce, Bureau of Export Administration for more information.

Patents: Implementations of the SHA-1 in this standard may be covered by U.S. and foreign patents.

Implementation Schedule: This standard becomes effective October 2, 1995.

Specifications: Federal Information Processing Standard (FIPS 180-1)

Secure Hash Standard (affixed).

Cross Index:

- a. FIPS PUB 46-2, Data Encryption Standard.
- b. FIPS PUB 73, Guidelines for Security of Computer Applications.
- c. FIPS PUB 140-1, Security Requirements for Cryptographic Modules.
- d. FIPS PUB 186, Digital Signature Standard.
- e. Federal Informations Resources Management Regulations (FIRMR) subpart 201.20.303, Standards, and subpart 201.39.1002, Federal Standards.

Objectives: The objectives of this standard are to:

- a. Specify the secure hash algorithm required for use with the Digital Signature Standard (FIPS 186) in the generation and verification of digital signatures;
- b. Specify the secure hash algorithm to be used whenever a secure hash algorithm is required for Federal applications; and
- c. Encourage the adoption and use of the specified secure hash algorithm by private and commercial organizations.

Qualifications: While it is the intent of this standard to specify a secure hash algorithm, conformance to this standard does not assure that a particular implementation is secure. The responsible authority in each agency or department shall assure that an overall implementation provides an acceptable level of security. This standard will be reviewed every five years in order to assess its adequacy.

Waiver Procedure: Under certain exceptional circumstances, the heads of Federal departments and agencies may approve waivers to Federal Information Processing Standards (FIPS). The head of such agency may redelegate such authority only to a senior official designated pursuant to section 3506(b) of Title 44, United States Code. Waiver shall be granted only when:

- a. Compliance with a standard would adversely affect the accomplishment of the mission of an operator of a Federal computer system; or
- b. Compliance with a standard would cause a major adverse financial impact on the operator which is not offset by Government-wide savings.

Agency heads may act upon a written waiver request containing the information detailed above. Agency heads may also act without a written waiver request when they determine that conditions for meeting the standard cannot be met. Agency heads may approve waivers only by a written decision which explains the basis on which the agency head made the required finding(s). A copy of each decision, with procurement sensitive or classified portions clearly identified, shall be sent to: National Institute of Standards and Technology;

ATTN: FIPS Waiver Decisions, Technology Building, Room B-154, Gaithersburg, MD 20899.

In addition, notice of each waiver granted and each delegation of authority to approve waivers shall be sent promptly to the Committee on Government Operations of the House of Representatives and the Committee on Government Affairs of the Senate and shall be published promptly in the Federal Register.

When the determination on a waiver applies to the procurement of equipment and/or services, a notice of the waiver determination must be published in the Commerce Business Daily as a part of the notice of solicitation for offers of an acquisition or, if the waiver determination is made after that notice is published, by amendment to such notice.

A copy of the waiver, any supporting documents, the document approving the waiver and any accompanying documents, with such deletions as the agency is authorized and decides to make under 5 United States Code Section 552(b), shall be part of the procurement documentation and retained by the agency.

Where to Obtain Copies of the Standard: Copies of this publication are for sale by the National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161. When ordering, refer to Federal Information Processing Standards Publication 180-1 (FIPSPUB180-1), and identify the title. When microfiche is desired, this should be specified. Prices are published by NTIS in current catalogs and other issuances. Payment may be made by check, money order, deposit account or charged to a credit card accepted by NTIS.

Federal Information

Processing Standards Publication 180-1

1995 April 17

Specifications for the

SECURE HASH STANDARD

1. INTRODUCTION

The Secure Hash Algorithm (SHA-1) is required for use with the Digital Signature Algorithm (DSA) as specified in the Digital Signature Standard (DSS) and whenever a secure hash algorithm is required for federal applica- tions. For a message of length < 2^64 bits, the SHA-1 produces a 160-bit condensed representation of the message called a message digest. The message digest is used during generation of a signature for the message. The SHA-1 is also used to compute a message digest for the received version of the message during the process of verifying the signature. Any change to the message in transit will, with very high probability, result in a different message digest, and the signature will fail to verify.

The SHA-1 is designed to have the following properties: it is computationally infeasible to find a message which corresponds to a given message digest, or to find two different messages which produce the same message digest.

2. BIT STRINGS AND INTEGERS

The following terminology related to bit strings and integers will be used:

- a. A hex digit is an element of the set {0, 1, ... , 9, A, ... , F}. A hex digit is the representation of a 4-bit string. Examples: 7 = 0111, A = 1010.
- b. A word equals a 32-bit string which may be represented as a sequence of 8 hex digits. To convert a word to 8 hex digits each 4-bit string is converted to its hex equivalent as described in (a) above. Example:
- 1010 0001 0000 0011 1111 1110 0010 0011 = A103FE23.
- c. An integer between 0 and 2^32 - 1 inclusive may be represented as a word.
- The least significant four bits of the integer are represented by the right-most hex digit of the word representation. Example: the integer 291 = 2^8+2^5+2^1+2^0 = 256+32+2+1 is represented by the hex word, 00000123.
- If z is an integer, 0 <= z < 2^64, then z = (2^32)x + y where 0 <= x < 2^32 and 0 <= y < 2^32. Since x and y can be represented as words X and Y, respectively, z can be represented as the pair of words (X,Y).
- d. block = 512-bit string. A block (e.g., B) may be represented as a sequence of 16 words.

3. OPERATIONS ON WORDS

The following logical operators will be applied to words:

- a. Bitwise logical word operations

X AND Y = bitwise logical "and" of X and Y.

X OR Y = bitwise logical "inclusive-or" of X and Y.

X XOR Y = bitwise logical "exclusive-or" of X and Y.

NOT X = bitwise logical "complement" of X.

Example:

01101100101110011101001001111011

XOR 01100101110000010110100110110111

= 00001001011110001011101111001100

b. The operation X + Y is defined as follows: words X and Y represent integers x and y, where 0 <= x < 2^32 and 0 <= y < 2^32. For positive integers n and m, let n mod m be the remainder upon dividing n by m.

Compute

z = (x + y) mod 2^32.

Then 0 <= z < 2^32. Convert z to a word, Z, and define Z = X + Y.

c. The circular left shift operation S^n(X), where X is a word and n is an integer with 0 <= n < 32, is defined by

S^n(X) = (X << n) OR (X >> 32-n).

In the above, X << n is obtained as follows: discard the left-most n bits of X and then pad the result with n zeroes on the right (the result will still be 32 bits). X >> n is obtained by discarding the right-most n bits of X and then padding the result with n zeroes on the left. Thus S^n(X) is equivalent to a circular shift of X by n positions to the left.

4. MESSAGE PADDING

The SHA-1 is used to compute a message digest for a message or data file that is provided as input. The message or data file should be considered to be a bit string. The length of the message is the number of bits in the message (the empty message has length 0). If the number of bits in a message is a multiple of 8, for compactness we can represent the message in hex. The purpose of message padding is to make the total length of a padded message a multiple of 512. The SHA-1 sequentially processes blocks of 512 bits when computing the message digest. The following specifies how this padding shall be performed. As a summary, a "1" followed by m "0"s followed by a 64-bit integer are appended to the end of the message to produce a padded message of length 512 * n. The 64-

bit integer is l, the length of the original message. The padded message is then processed by the SHA-1 as n 512-bit blocks.

Suppose a message has length l < 2^64. Before it is input to the SHA-1, the message is padded on the right as follows:

- a. "1" is appended. Example: if the original message is "01010000", this is padded to "010100001".
- b. "0"s are appended. The number of "0"s will depend on the original length of the message. The last 64 bits of the last 512-bit block are reserved for the length l of the original message.
- Example: Suppose the original message is the bit string

01100001 01100010 01100011 01100100 01100101.

After step (a) this gives

01100001 01100010 01100011 01100100 01100101 1.

Since l = 40, the number of bits in the above is 41 and 407 "0"s are appended, making the total now 448. This gives (in hex)

- 61626364 65800000 00000000 00000000
- 00000000 00000000 00000000 00000000
- 00000000 00000000 00000000 00000000
- 00000000 00000000.
- c. Obtain the 2-word representation of l, the number of bits in the original message. If l < 2^32 then the first word is all zeroes. Append these two words to the padded message.
- Example: Suppose the original message is as in (b). Then l = 40 (note
- that l is computed before any padding). The two-word representation of
- 40 is hex 00000000 00000028. Hence the final padded message is hex
  - 61626364 65800000 00000000 00000000
  - 00000000 00000000 00000000 00000000
  - 00000000 00000000 00000000 00000000
  - 00000000 00000000 00000000 00000028.

The padded message will contain 16 * n words for some n > 0. The padded message is regarded as a sequence of n blocks M(1) , M(2), ... , M(n), where each M(i) contains 16 words and M(1) contains the first characters (or bits) of the message.

5. FUNCTIONS USED

A sequence of logical functions f(0), f(1),..., f(79) is used in the SHA-1.

Each f(t), 0 <= t <= 79, operates on three 32-bit words B, C, D and produces

a 32-bit word as output. f(t;B,C,D) is defined as follows:

for words B, C, D,

f(t;B,C,D) = (B AND C) OR ((NOT B) AND D) (0 <= t <= 19)

f(t;B,C,D) = B XOR C XOR D (20 <= t <= 39)

f(t;B,C,D) = (B AND C) OR (B AND D) OR (C AND D) (40 <= t <= 59)

f(t;B,C,D) = B XOR C XOR D (60 <= t <= 79).

6. CONSTANTS USED

A sequence of constant words K(0), K(1), ... , K(79) is used in the SHA-1.

In hex these are given by

K(t) = 5A827999 (0 <= t <= 19)

K(t) = 6ED9EBA1 (20 <= t <= 39)

K(t) = 8F1BBCDC (40 <= t <= 59)

K(t) = CA62C1D6 (60 <= t <= 79).

7. COMPUTING THE MESSAGE DIGEST

The message digest is computed using the final padded message. The computation uses two buffers, each consisting of five 32-bit words, and a sequence of eighty 32-bit words. The words of the first 5-word buffer are labeled A,B,C,D,E. The words of the second 5-word buffer are labeled H0, H1, H2, H3, H4. The words of the 80-word sequence are labeled W(0), W(1),..., W(79). A single word buffer TEMP is also employed.

To generate the message digest, the 16-word blocks M(1), M(2),..., M(n) defined in Section 4 are processed in order. The processing of each M(i) involves 80 steps.

Before processing any blocks, the H's are initialized as follows:

in hex,

H0 = 67452301

H1 = EFCDAB89

H2 = 98BADCFE

H3 = 10325476

H4 = C3D2E1F0.

Now M(1), M(2), ... , M(n) are processed. To process M(i), we proceed as follows:

- a. Divide M(i) into 16 words W(0), W(1), ... , W(15), where W(0) is the left-most word.
- b. For t = 16 to 79 let
- W(t) = S^1(W(t-3) XOR W(t-8) XOR W(t-14) XOR W(t-16)).
- c. Let A = H0, B = H1, C = H2, D = H3, E = H4.
- d. For t = 0 to 79 do
- TEMP = S^5(A) + f(t;B,C,D) + E + W(t) + K(t);

E = D; D = C; C = S^30(B); B = A; A = TEMP;

- e. Let H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D, H4 = H4 + E.
- After processing M(n), the message digest is the 160-bit string

represented by the 5 words

H0 H1 H2 H3 H4.

8. ALTERNATE METHOD OF COMPUTATION

The above assumes that the sequence W(0), ... , W(79) is implemented as an array of eighty 32-bit words. This is efficient from the standpoint of minimization of execution time, since the addresses of W(t-3), ... ,W(t-16) in step (b) are easily computed. If space is at a premium, an alternative is to regard { W(t) } as a circular queue, which may be implemented using an array of sixteen 32-bit words W[0], ... W[15]. In this case, in hex let MASK = 0000000F. Then processing of M(i) is as follows:

- a. Divide M(i) into 16 words W[0], ... , W[15], where W[0] is the left-most word.
- b. Let A = H0, B = H1, C = H2, D = H3, E = H4.
- c. For t = 0 to 79 do
- s = t AND MASK;
- if (t >= 16) W[s] = S^1(W[(s + 13) AND MASK] XOR W[(s + 8) AND MASK]

XOR W[(s + 2) AND MASK] XOR W[s]);

TEMP = S^5(A) + f(t;B,C,D) + E + W[s] + K(t);

- E = D; D = C; C = S^30(B); B = A; A = TEMP; d. Let H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D, H4 = H4 + E.

9. COMPARISON OF METHODS

The methods of Sections 7 and 8 yield the same message digest. Although using the method of Section 8 saves sixty-four 32-bit words of storage, it is likely to lengthen execution time due to the increased complexity of the address computations for the { W[t] } in step ©. Other computation methods which give identical results may be implemented in conformance with the standard.

APPENDIX A. A SAMPLE MESSAGE AND ITS MESSAGE DIGEST

This appendix is for informational purposes only and is not required to meet the standard.

Let the message be the ASCII binary-coded form of "abc", i.e.,

01100001 01100010 01100011.

This message has length l = 24. In step (a) of Section 4, we append "1". In step (b) we append 423 "0"s. In step © we append hex 00000000 00000018, the 2-word representation of 24. Thus the final padded message consists of one block, so that n = 1 in the notation of Section 4.

The initial hex values of {Hi} are

H0 = 67452301 H1 = EFCDAB89 H2 = 98BADCFE

H3 = 10325476

H4 = C3D2E1F0.

Start processing block 1. The words of block 1 are

W[0] = 61626380

- W[1] = 00000000
- W[2] = 00000000
- W[3] = 00000000
- W[4] = 00000000
- W[5] = 00000000
- W[6] = 00000000
- W[7] = 00000000

W[8] = 00000000
W[9] = 00000000
W[10] = 00000000
W[11] = 00000000
W[12] = 00000000
W[13] = 00000000
W[14] = 00000000
W[15] = 00000018.

The hex values of A,B,C,D,E after pass t of the "for t = 0 to 79" loop

(step (d) of Section 7 or step © of Section 8) are

A B C D E

t = 0: 0116FC33 67452301 7BF36AE2 98BADCFE 10325476

t = 1: 8990536D 0116FC33 59D148C0 7BF36AE2 98BADCFE

t = 2: A1390F08 8990536D C045BF0C 59D148C0 7BF36AE2

t = 3: CDD8E11B A1390F08 626414DB C045BF0C 59D148C0

t = 4: CFD499DE CDD8E11B 284E43C2 626414DB C045BF0C

t = 5: 3FC7CA40 CFD499DE F3763846 284E43C2 626414DB

t = 6: 993E30C1 3FC7CA40 B3F52677 F3763846 284E43C2

t = 7: 9E8C07D4 993E30C1 0FF1F290 B3F52677 F3763846

t = 8: 4B6AE328 9E8C07D4 664F8C30 0FF1F290 B3F52677

t = 9: 8351F929 4B6AE328 27A301F5 664F8C30 0FF1F290

t = 10: FBDA9E89 8351F929 12DAB8CA 27A301F5 664F8C30

t = 11: 63188FE4 FBDA9E89 60D47E4A 12DAB8CA 27A301F5

t = 12: 4607B664 63188FE4 7EF6A7A2 60D47E4A 12DAB8CA

t = 13: 9128F695 4607B664 18C623F9 7EF6A7A2 60D47E4A

t = 14: 196BEE77 9128F695 1181ED99 18C623F9 7EF6A7A2

t = 15: 20BDD62F 196BEE77 644A3DA5 1181ED99 18C623F9

t = 16: 4E925823 20BDD62F C65AFB9D 644A3DA5 1181ED99

t = 17: 82AA6728 4E925823 C82F758B C65AFB9D 644A3DA5

t = 18: DC64901D 82AA6728 D3A49608 C82F758B C65AFB9D

t = 19: FD9E1D7D DC64901D 20AA99CA D3A49608 C82F758B

t = 20: 1A37B0CA FD9E1D7D 77192407 20AA99CA D3A49608

t = 21: 33A23BFC 1A37B0CA 7F67875F 77192407 20AA99CA

t = 22: 21283486 33A23BFC 868DEC32 7F67875F 77192407

t = 23: D541F12D 21283486 0CE88EFF 868DEC32 7F67875F

t = 24: C7567DC6 D541F12D 884A0D21 0CE88EFF 868DEC32

t = 25: 48413BA4 C7567DC6 75507C4B 884A0D21 0CE88EFF

t = 26: BE35FBD5 48413BA4 B1D59F71 75507C4B 884A0D21

t = 27: 4AA84D97 BE35FBD5 12104EE9 B1D59F71 75507C4B

t = 28: 8370B52E 4AA84D97 6F8D7EF5 12104EE9 B1D59F71

t = 29: C5FBAF5D 8370B52E D2AA1365 6F8D7EF5 12104EE9

t = 30: 1267B407 C5FBAF5D A0DC2D4B D2AA1365 6F8D7EF5

t = 31: 3B845D33 1267B407 717EEBD7 A0DC2D4B D2AA1365

t = 32: 046FAA0A 3B845D33 C499ED01 717EEBD7 A0DC2D4B

t = 33: 2C0EBC11 046FAA0A CEE1174C C499ED01 717EEBD7

t = 34: 21796AD4 2C0EBC11 811BEA82 CEE1174C C499ED01

t = 35: DCBBB0CB 21796AD4 4B03AF04 811BEA82 CEE1174C

t = 36: 0F511FD8 DCBBB0CB 085E5AB5 4B03AF04 811BEA82

t = 37: DC63973F 0F511FD8 F72EEC32 085E5AB5 4B03AF04

t = 38: 4C986405 DC63973F 03D447F6 F72EEC32 085E5AB5

t = 39: 32DE1CBA 4C986405 F718E5CF 03D447F6 F72EEC32

t = 40: FC87DEDF 32DE1CBA 53261901 F718E5CF 03D447F6

t = 41: 970A0D5C FC87DEDF 8CB7872E 53261901 F718E5CF

t = 42: 7F193DC5 970A0D5C FF21F7B7 8CB7872E 53261901

t = 43: EE1B1AAF 7F193DC5 25C28357 FF21F7B7 8CB7872E

t = 44: 40F28E09 EE1B1AAF 5FC64F71 25C28357 FF21F7B7

t = 45: 1C51E1F2 40F28E09 FB86C6AB 5FC64F71 25C28357

t = 46: A01B846C 1C51E1F2 503CA382 FB86C6AB 5FC64F71

t = 47: BEAD02CA A01B846C 8714787C 503CA382 FB86C6AB

t = 48: BAF39337 BEAD02CA 2806E11B 8714787C 503CA382

t = 49: 120731C5 BAF39337 AFAB40B2 2806E11B 8714787C

t = 50: 641DB2CE 120731C5 EEBCE4CD AFAB40B2 2806E11B

t = 51: 3847AD66 641DB2CE 4481CC71 EEBCE4CD AFAB40B2

t = 52: E490436D 3847AD66 99076CB3 4481CC71 EEBCE4CD

t = 53: 27E9F1D8 E490436D 8E11EB59 99076CB3 4481CC71

t = 54: 7B71F76D 27E9F1D8 792410DB 8E11EB59 99076CB3

t = 55: 5E6456AF 7B71F76D 09FA7C76 792410DB 8E11EB59

t = 56: C846093F 5E6456AF 5EDC7DDB 09FA7C76 792410DB

t = 57: D262FF50 C846093F D79915AB 5EDC7DDB 09FA7C76

t = 58: 09D785FD D262FF50 F211824F D79915AB 5EDC7DDB

t = 59: 3F52DE5A 09D785FD 3498BFD4 F211824F D79915AB

t = 60: D756C147 3F52DE5A 4275E17F 3498BFD4 F211824F

t = 61: 548C9CB2 D756C147 8FD4B796 4275E17F 3498BFD4

t = 62: B66C020B 548C9CB2 F5D5B051 8FD4B796 4275E17F

t = 63: 6B61C9E1 B66C020B 9523272C F5D5B051 8FD4B796

t = 64: 19DFA7AC 6B61C9E1 ED9B0082 9523272C F5D5B051

t = 65: 101655F9 19DFA7AC 5AD87278 ED9B0082 9523272C

t = 66: 0C3DF2B4 101655F9 0677E9EB 5AD87278 ED9B0082

t = 67: 78DD4D2B 0C3DF2B4 4405957E 0677E9EB 5AD87278

t = 68: 497093C0 78DD4D2B 030F7CAD 4405957E 0677E9EB

t = 69: 3F2588C2 497093C0 DE37534A 030F7CAD 4405957E

t = 70: C199F8C7 3F2588C2 125C24F0 DE37534A 030F7CAD

t = 71: 39859DE7 C199F8C7 8FC96230 125C24F0 DE37534A

t = 72: EDB42DE4 39859DE7 F0667E31 8FC96230 125C24F0

t = 73: 11793F6F EDB42DE4 CE616779 F0667E31 8FC96230

t = 74: 5EE76897 11793F6F 3B6D0B79 CE616779 F0667E31

t = 75: 63F7DAB7 5EE76897 C45E4FDB 3B6D0B79 CE616779

t = 76: A079B7D9 63F7DAB7 D7B9DA25 C45E4FDB 3B6D0B79

t = 77: 860D21CC A079B7D9 D8FDF6AD D7B9DA25 C45E4FDB

t = 78: 5738D5E1 860D21CC 681E6DF6 D8FDF6AD D7B9DA25

t = 79: 42541B35 5738D5E1 21834873 681E6DF6 D8FDF6AD.

Block 1 has been processed. The values of {Hi} are

H0 = 67452301 + 42541B35 = A9993E36 H1 = EFCDAB89 + 5738D5E1 = 4706816A H2 = 98BADCFE + 21834873 = BA3E2571 H3 = 10325476 + 681E6DF6 = 7850C26C H4 = C3D2E1F0 + D8FDF6AD = 9CD0D89D. Message digest = A9993E36 4706816A BA3E2571 7850C26C 9CD0D89D

APPENDIX B. A SECOND SAMPLE MESSAGE AND ITS MESSAGE DIGEST

This appendix is for informational purposes only and is not required to meet the standard.

Let the message be the binary-coded form (cf. Appendix A) of the ASCII string "abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq".

Since each of the 56 characters is converted to 8 bits, the length of the message is l = 448. In step (a) of Section 4, we append "1". In step (b) we append 511 "0"s. In step © we append the 2-word representation of 448, i.e., hex 00000000 000001C0. This gives n = 2.

The initial hex values of {Hi} are

H0 = 67452301 H1 = EFCDAB89 H2 = 98BADCFE H3 = 10325476 H4 = C3D2E1F0. Start processing block 1. The words of block 1 are

- W[0] = 61626364 W[1] = 62636465
- W[2] = 63646566
- W[3] = 64656667
- W[4] = 65666768
- W[5] = 66676869
- W[6] = 6768696A
- W[7] = 68696A6B
- W[8] = 696A6B6C
- W[9] = 6A6B6C6D
- W[10] = 6B6C6D6E
- W[11] = 6C6D6E6F
- W[12] = 6D6E6F70
- W[13] = 6E6F7071
- W[14] = 80000000
- W[15] = 00000000.

The hex values of A,B,C,D,E after pass t of the "for t = 0 to 79" loop (step

(d) of Section 7 or step © of Section 8) are

A B C D E

t = 0: 0116FC17 67452301 7BF36AE2 98BADCFE 10325476

t = 1: EBF3B452 0116FC17 59D148C0 7BF36AE2 98BADCFE

t = 2: 5109913A EBF3B452 C045BF05 59D148C0 7BF36AE2

t = 3: 2C4F6EAC 5109913A BAFCED14 C045BF05 59D148C0

t = 4: 33F4AE5B 2C4F6EAC 9442644E BAFCED14 C045BF05

t = 5: 96B85189 33F4AE5B 0B13DBAB 9442644E BAFCED14

t = 6: DB04CB58 96B85189 CCFD2B96 0B13DBAB 9442644E

t = 7: 45833F0F DB04CB58 65AE1462 CCFD2B96 0B13DBAB

t = 8: C565C35E 45833F0F 36C132D6 65AE1462 CCFD2B96

t = 9: 6350AFDA C565C35E D160CFC3 36C132D6 65AE1462

t = 10: 8993EA77 6350AFDA B15970D7 D160CFC3 36C132D6

t = 11: E19ECAA2 8993EA77 98D42BF6 B15970D7 D160CFC3

t = 12: 8603481E E19ECAA2 E264FA9D 98D42BF6 B15970D7

t = 13: 32F94A85 8603481E B867B2A8 E264FA9D 98D42BF6

t = 14: B2E7A8BE 32F94A85 A180D207 B867B2A8 E264FA9D

t = 15: 42637E39 B2E7A8BE 4CBE52A1 A180D207 B867B2A8

t = 16: 6B068048 42637E39 ACB9EA2F 4CBE52A1 A180D207

t = 17: 426B9C35 6B068048 5098DF8E ACB9EA2F 4CBE52A1

t = 18: 944B1BD1 426B9C35 1AC1A012 5098DF8E ACB9EA2F

t = 19: 6C445652 944B1BD1 509AE70D 1AC1A012 5098DF8E

t = 20: 95836DA5 6C445652 6512C6F4 509AE70D 1AC1A012

t = 21: 09511177 95836DA5 9B111594 6512C6F4 509AE70D

t = 22: E2B92DC4 09511177 6560DB69 9B111594 6512C6F4

t = 23: FD224575 E2B92DC4 C254445D 6560DB69 9B111594

t = 24: EEB82D9A FD224575 38AE4B71 C254445D 6560DB69

t = 25: 5A142C1A EEB82D9A 7F48915D 38AE4B71 C254445D

t = 26: 2972F7C7 5A142C1A BBAE0B66 7F48915D 38AE4B71

t = 27: D526A644 2972F7C7 96850B06 BBAE0B66 7F48915D

t = 28: E1122421 D526A644 CA5CBDF1 96850B06 BBAE0B66

t = 29: 05B457B2 E1122421 3549A991 CA5CBDF1 96850B06

t = 30: A9C84BEC 05B457B2 78448908 3549A991 CA5CBDF1

t = 31: 52E31F60 A9C84BEC 816D15EC 78448908 3549A991

t = 32: 5AF3242C 52E31F60 2A7212FB 816D15EC 78448908

t = 33: 31C756A9 5AF3242C 14B8C7D8 2A7212FB 816D15EC

t = 34: E9AC987C 31C756A9 16BCC90B 14B8C7D8 2A7212FB

t = 35: AB7C32EE E9AC987C 4C71D5AA 16BCC90B 14B8C7D8

t = 36: 5933FC99 AB7C32EE 3A6B261F 4C71D5AA 16BCC90B

t = 37: 43F87AE9 5933FC99 AADF0CBB 3A6B261F 4C71D5AA

t = 38: 24957F22 43F87AE9 564CFF26 AADF0CBB 3A6B261F

t = 39: ADEB7478 24957F22 50FE1EBA 564CFF26 AADF0CBB

t = 40: D70E5010 ADEB7478 89255FC8 50FE1EBA 564CFF26

t = 41: 79BCFB08 D70E5010 2B7ADD1E 89255FC8 50FE1EBA

t = 42: F9BCB8DE 79BCFB08 35C39404 2B7ADD1E 89255FC8

t = 43: 633E9561 F9BCB8DE 1E6F3EC2 35C39404 2B7ADD1E

t = 44: 98C1EA64 633E9561 BE6F2E37 1E6F3EC2 35C39404

t = 45: C6EA241E 98C1EA64 58CFA558 BE6F2E37 1E6F3EC2

t = 46: A2AD4F02 C6EA241E 26307A99 58CFA558 BE6F2E37

t = 47: C8A69090 A2AD4F02 B1BA8907 26307A99 58CFA558

t = 48: 88341600 C8A69090 A8AB53C0 B1BA8907 26307A99

t = 49: 7E846F58 88341600 3229A424 A8AB53C0 B1BA8907

t = 50: 86E358BA 7E846F58 220D0580 3229A424 A8AB53C0

t = 51: 8D2E76C8 86E358BA 1FA11BD6 220D0580 3229A424

t = 52: CE892E10 8D2E76C8 A1B8D62E 1FA11BD6 220D0580

t = 53: EDEA95B1 CE892E10 234B9DB2 A1B8D62E 1FA11BD6

t = 54: 36D1230A EDEA95B1 33A24B84 234B9DB2 A1B8D62E

t = 55: 776C3910 36D1230A 7B7AA56C 33A24B84 234B9DB2

t = 56: A681B723 776C3910 8DB448C2 7B7AA56C 33A24B84

t = 57: AC0A794F A681B723 1DDB0E44 8DB448C2 7B7AA56C

t = 58: F03D3782 AC0A794F E9A06DC8 1DDB0E44 8DB448C2

t = 59: 9EF775C3 F03D3782 EB029E53 E9A06DC8 1DDB0E44

t = 60: 36254B13 9EF775C3 BC0F4DE0 EB029E53 E9A06DC8

t = 61: 4080D4DC 36254B13 E7BDDD70 BC0F4DE0 EB029E53

t = 62: 2BFAF7A8 4080D4DC CD8952C4 E7BDDD70 BC0F4DE0

t = 63: 513F9CA0 2BFAF7A8 10203537 CD8952C4 E7BDDD70

t = 64: E5895C81 513F9CA0 0AFEBDEA 10203537 CD8952C4

t = 65: 1037D2D5 E5895C81 144FE728 0AFEBDEA 10203537

t = 66: 14A82DA9 1037D2D5 79625720 144FE728 0AFEBDEA

t = 67: 6D17C9FD 14A82DA9 440DF4B5 79625720 144FE728

t = 68: 2C7B07BD 6D17C9FD 452A0B6A 440DF4B5 79625720

t = 69: FDF6EFFF 2C7B07BD 5B45F27F 452A0B6A 440DF4B5

t = 70: 112B96E3 FDF6EFFF 4B1EC1EF 5B45F27F 452A0B6A

t = 71: 84065712 112B96E3 FF7DBBFF 4B1EC1EF 5B45F27F

t = 72: AB89FB71 84065712 C44AE5B8 FF7DBBFF 4B1EC1EF

t = 73: C5210E35 AB89FB71 A10195C4 C44AE5B8 FF7DBBFF

t = 74: 352D9F4B C5210E35 6AE27EDC A10195C4 C44AE5B8

t = 75: 1A0E0E0A 352D9F4B 7148438D 6AE27EDC A10195C4

t = 76: D0D47349 1A0E0E0A CD4B67D2 7148438D 6AE27EDC

t = 77: AD38620D D0D47349 86838382 CD4B67D2 7148438D

t = 78: D3AD7C25 AD38620D 74351CD2 86838382 CD4B67D2

t = 79: 8CE34517 D3AD7C25 6B4E1883 74351CD2 86838382.

Block 1 has been processed. The values of {Hi} are

H0 = 67452301 + 8CE34517 = F4286818 H1 = EFCDAB89 + D3AD7C25 = C37B27AE H2 = 98BADCFE + 6B4E1883 = 0408F581 H3 = 10325476 + 74351CD2 = 84677148 H4 = C3D2E1F0 + 86838382 = 4A566572. Start processing block 2. The words of block 2 are

- W[0] = 00000000 W[1] = 00000000
- W[2] = 00000000
- W[3] = 00000000
- W[4] = 00000000
- W[5] = 00000000
- W[6] = 00000000
- W[7] = 00000000
- W[8] = 00000000
- W[9] = 00000000
- W[10] = 00000000
- W[11] = 00000000
- W[12] = 00000000
- W[13] = 00000000
- W[14] = 00000000
- W[15] = 000001C0.

The hex values of A,B,C,D,E after pass t of the for "t = 0 to 79" loop

(step (d) of Section 7 or step © of Section 8) are

A B C D E

t = 0: 2DF257E9 F4286818 B0DEC9EB 0408F581 84677148

t = 1: 4D3DC58F 2DF257E9 3D0A1A06 B0DEC9EB 0408F581

t = 2: C352BB05 4D3DC58F 4B7C95FA 3D0A1A06 B0DEC9EB

t = 3: EEF743C6 C352BB05 D34F7163 4B7C95FA 3D0A1A06

t = 4: 41E34277 EEF743C6 70D4AEC1 D34F7163 4B7C95FA

t = 5: 5443915C 41E34277 BBBDD0F1 70D4AEC1 D34F7163

t = 6: E7FA0377 5443915C D078D09D BBBDD0F1 70D4AEC1

t = 7: C6946813 E7FA0377 1510E457 D078D09D BBBDD0F1

t = 8: FDDE1DE1 C6946813 F9FE80DD 1510E457 D078D09D

t = 9: B8538ACA FDDE1DE1 F1A51A04 F9FE80DD 1510E457

t = 10: 6BA94F63 B8538ACA 7F778778 F1A51A04 F9FE80DD

t = 11: 43A2792F 6BA94F63 AE14E2B2 7F778778 F1A51A04

t = 12: FECD7BBF 43A2792F DAEA53D8 AE14E2B2 7F778778

t = 13: A2604CA8 FECD7BBF D0E89E4B DAEA53D8 AE14E2B2

t = 14: 258B0BAA A2604CA8 FFB35EEF D0E89E4B DAEA53D8

t = 15: D9772360 258B0BAA 2898132A FFB35EEF D0E89E4B

t = 16: 5507DB6E D9772360 8962C2EA 2898132A FFB35EEF

t = 17: A51B58BC 5507DB6E 365DC8D8 8962C2EA 2898132A

t = 18: C2EB709F A51B58BC 9541F6DB 365DC8D8 8962C2EA

t = 19: D8992153 C2EB709F 2946D62F 9541F6DB 365DC8D8

t = 20: 37482F5F D8992153 F0BADC27 2946D62F 9541F6DB

t = 21: EE8700BD 37482F5F F6264854 F0BADC27 2946D62F

t = 22: 9AD594B9 EE8700BD CDD20BD7 F6264854 F0BADC27

t = 23: 8FBAA5B9 9AD594B9 7BA1C02F CDD20BD7 F6264854

t = 24: 88FB5867 8FBAA5B9 66B5652E 7BA1C02F CDD20BD7

t = 25: EEC50521 88FB5867 63EEA96E 66B5652E 7BA1C02F

t = 26: 50BCE434 EEC50521 E23ED619 63EEA96E 66B5652E

t = 27: 5C416DAF 50BCE434 7BB14148 E23ED619 63EEA96E

t = 28: 2429BE5F 5C416DAF 142F390D 7BB14148 E23ED619

t = 29: 0A2FB108 2429BE5F D7105B6B 142F390D 7BB14148

t = 30: 17986223 0A2FB108 C90A6F97 D7105B6B 142F390D

t = 31: 8A4AF384 17986223 028BEC42 C90A6F97 D7105B6B

t = 32: 6B629993 8A4AF384 C5E61888 028BEC42 C90A6F97

t = 33: F15F04F3 6B629993 2292BCE1 C5E61888 028BEC42

t = 34: 295CC25B F15F04F3 DAD8A664 2292BCE1 C5E61888

t = 35: 696DA404 295CC25B FC57C13C DAD8A664 2292BCE1

t = 36: CEF5AE12 696DA404 CA573096 FC57C13C DAD8A664

t = 37: 87D5B80C CEF5AE12 1A5B6901 CA573096 FC57C13C

t = 38: 84E2A5F2 87D5B80C B3BD6B84 1A5B6901 CA573096

t = 39: 03BB6310 84E2A5F2 21F56E03 B3BD6B84 1A5B6901

t = 40: C2D8F75F 03BB6310 A138A97C 21F56E03 B3BD6B84

t = 41: BFB25768 C2D8F75F 00EED8C4 A138A97C 21F56E03

t = 42: 28589152 BFB25768 F0B63DD7 00EED8C4 A138A97C

t = 43: EC1D3D61 28589152 2FEC95DA F0B63DD7 00EED8C4

t = 44: 3CAED7AF EC1D3D61 8A162454 2FEC95DA F0B63DD7

t = 45: C3D033EA 3CAED7AF 7B074F58 8A162454 2FEC95DA

t = 46: 7316056A C3D033EA CF2BB5EB 7B074F58 8A162454

t = 47: 46F93B68 7316056A B0F40CFA CF2BB5EB 7B074F58

t = 48: DC8E7F26 46F93B68 9CC5815A B0F40CFA CF2BB5EB

t = 49: 850D411C DC8E7F26 11BE4EDA 9CC5815A B0F40CFA

t = 50: 7E4672C0 850D411C B7239FC9 11BE4EDA 9CC5815A

t = 51: 89FBD41D 7E4672C0 21435047 B7239FC9 11BE4EDA

t = 52: 1797E228 89FBD41D 1F919CB0 21435047 B7239FC9

t = 53: 431D65BC 1797E228 627EF507 1F919CB0 21435047

t = 54: 2BDBB8CB 431D65BC 05E5F88A 627EF507 1F919CB0

t = 55: 6DA72E7F 2BDBB8CB 10C7596F 05E5F88A 627EF507

t = 56: A8495A9B 6DA72E7F CAF6EE32 10C7596F 05E5F88A

t = 57: E785655A A8495A9B DB69CB9F CAF6EE32 10C7596F

t = 58: 5B086C42 E785655A EA1256A6 DB69CB9F CAF6EE32

t = 59: A65818F7 5B086C42 B9E15956 EA1256A6 DB69CB9F

t = 60: 7AAB101B A65818F7 96C21B10 B9E15956 EA1256A6

t = 61: 93614C9C 7AAB101B E996063D 96C21B10 B9E15956

t = 62: F66D9BF4 93614C9C DEAAC406 E996063D 96C21B10

t = 63: D504902B F66D9BF4 24D85327 DEAAC406 E996063D

t = 64: 60A9DA62 D504902B 3D9B66FD 24D85327 DEAAC406

t = 65: 8B687819 60A9DA62 F541240A 3D9B66FD 24D85327

t = 66: 083E90C3 8B687819 982A7698 F541240A 3D9B66FD

t = 67: F6226BBF 083E90C3 62DA1E06 982A7698 F541240A

t = 68: 76C0563B F6226BBF C20FA430 62DA1E06 982A7698

t = 69: 989DD165 76C0563B FD889AEF C20FA430 62DA1E06

t = 70: 8B2C7573 989DD165 DDB0158E FD889AEF C20FA430

t = 71: AE1B8E7B 8B2C7573 66277459 DDB0158E FD889AEF

t = 72: CA1840DE AE1B8E7B E2CB1D5C 66277459 DDB0158E

t = 73: 16F3BABB CA1840DE EB86E39E E2CB1D5C 66277459

t = 74: D28D83AD 16F3BABB B2861037 EB86E39E E2CB1D5C

t = 75: 6BC02DFE D28D83AD C5BCEEAE B2861037 EB86E39E

t = 76: D3A6E275 6BC02DFE 74A360EB C5BCEEAE B2861037

t = 77: DA955482 D3A6E275 9AF00B7F 74A360EB C5BCEEAE

t = 78: 58C0AAC0 DA955482 74E9B89D 9AF00B7F 74A360EB

t = 79: 906FD62C 58C0AAC0 B6A55520 74E9B89D 9AF00B7F.

Block 2 has been processed. The values of {Hi} are

H0 = F4286818 + 906FD62C = 84983E44 H1 = C37B27AE + 58C0AAC0 = 1C3BD26E H2 = 0408F581 + B6A55520 = BAAE4AA1 H3 = 84677148 + 74E9B89D = F95129E5 H4 = 4A566572 + 9AF00B7F = E54670F1. Message digest = 84983E44 1C3BD26E BAAE4AA1 F95129E5 E54670F1

APPENDIX C. A THIRD SAMPLE MESSAGE AND ITS MESSAGE DIGEST

This appendix is for informational purposes only and is not required to meet the standard.

Let the message be the binary-coded form of the ASCII string which consists of 1,000,000 repetitions of "a".

Message digest = 34AA973C D4C4DAA4 F61EEB2B DBAD2731 6534016F