FIPS PUB 180-4

FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION
联邦信息处理标准出版物

Secure Hash Standard (SHS)

安全哈希标准

CATEGORY: COMPUTER SECURITY SUBCATEGORY: CRYPTOGRAPHY
类别: 计算机安全 子类别: 密码学

Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8900

August 2015

U.S. Department of Commerce
Penny Pritzker, Secretary

National Institute of Standards and Technology
Willie E. May, Under Secretary for Standards and Technology and Director

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 the Federal Information Security Management Act (FISMA) of 2002.

Comments concerning FIPS publications are welcomed and should be addressed to the Director, Information Technology Laboratory, National Institute of Standards and Technology, 100 Bureau Drive, Stop 8900, Gaithersburg, MD 20899-8900.

Charles H. Romine, Director
Information Technology Laboratory

前言

美国国家标准与技术研究所 (***National Institute of Standards and Technology***, NIST) 的《联邦信息处理标准出版系列》是根据 2002 年《联邦信息安全管理法》(***Federal Information Security Management Act***, FISMA) 的规定，通过与颁布的标准和指南的官方系列出版物。

欢迎对联邦信息处理标准 (***Federal Information Processing Standards***, FIPS) 出版物发表意见，并将意见发送至美国国家标准与技术研究所信息技术实验室主任，地址：100 Bureau Drive, Stop 8900, Gaithersburg, MD 20899-8900.

Charles H. Romine, Director
Information Technology Laboratory

Abstract

This standard specifies hash algorithms that can be used to generate digests of messages. The digests are used to detect whether messages have been changed since the digests were generated.

Key words: computer security, cryptography, message digest, hash function, hash algorithm, Federal Information Processing Standards, Secure Hash Standard.

摘要

本标准规定了可用于生成消息摘要(***Message digest***)的哈希算法(***Hash algorithms***)。摘要用于检测自摘要生成以来消息是否已更改。

关键词：计算机安全、密码学、消息摘要、哈希函数、哈希算法、联邦信息处理标准、安全哈希标准。

Federal Information
Processing Standards Publication 180-4

August 2015

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 5131 of the Information Technology Management Reform Act of 1996 (Public Law 104-106), and the Computer Security Act of 1987 (Public Law 100-235).

1. **Name of Standard**: Secure Hash Standard (SHS) (FIPS PUB 180-4).
2. **Category of Standard**: Computer Security Standard, Cryptography.
3. **Explanation**: This Standard specifies secure hash algorithms - SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256 - for computing a condensed representation of electronic data (message). When a message of any length less than 264 bits (for SHA-1, SHA-224 and SHA-256) or less than 2128 bits (for SHA-384, SHA-512, SHA-512/224 and SHA-512/256) is input to a hash algorithm, the result is an output called a message digest. The message digests range in length from 160 to 512 bits, depending on the algorithm. Secure hash algorithms are typically used with other cryptographic algorithms, such as digital signature algorithms and keyed-hash message authentication codes, or in the generation of random numbers (bits).

   The hash algorithms specified in this Standard are called secure because, for a given algorithm, it is computationally infeasible 1) to find a message that corresponds to a given message digest, or 2) to find two different messages that produce the same message digest. Any change to a message will, with a very high probability, result in a different message digest. This will result in a verification failure when the secure hash algorithm is used with a digital signature algorithm or a keyed-hash message authentication algorithm.

   This Standard supersedes FIPS 180-3 [FIPS 180-3].
4. **Approving Authority**: Secretary of Commerce.
5. **Maintenance Agency**: U.S. Department of Commerce, National Institute of Standards and Technology (NIST), Information Technology Laboratory (ITL).
6. **Applicability**: This Standard is applicable to all Federal departments and agencies for the protection of sensitive unclassified information that is not subject to Title 10 United States Code Section 2315 (10 USC 2315) and that is not within a national

security system as defined in Title 40 United States Code Section 11103(a)(1) (40 USC 11103(a)(1)). Either this Standard or Federal Information Processing Standard (FIPS) 202 must be implemented wherever a secure hash algorithm is required for Federal applications, including as a component within other cryptographic algorithms and protocols. This Standard may be adopted and used by non-Federal Government organizations.

7. Specifications: Federal Information Processing Standard (FIPS) 180-4, Secure Hash Standard (SHS) (affixed).

8. Implementations: The secure hash algorithms specified herein may be implemented in software, firmware, hardware or any combination thereof. Only algorithm implementations that are validated by NIST will be considered as complying with this standard. Information about the validation program can be obtained at http://csrc.nist.gov/groups/STM/index.html.

9. Implementation Schedule: Guidance regarding the testing and validation to FIPS 180-4 and its relationship to FIPS 140-2 can be found in IG 1.10 of the Implementation Guidance for FIPS PUB 140-2 and the Cryptographic Module Validation Program at http://csrc.nist.gov/groups/STM/cmvp/index.html.

10. Patents: Implementations of the secure hash algorithms in this standard may be covered by U.S. or foreign patents.

11. Export Control: Certain cryptographic devices and technical data regarding them are subject to Federal export controls. Exports of cryptographic modules implementing this standard and technical data regarding them must comply with these Federal regulations and be licensed by the Bureau of Export Administration of the U.S. Department of Commerce. Information about export regulations is available at: http://www.bis.doc.gov/index.htm.

12. Qualifications: While it is the intent of this Standard to specify general security requirements for generating a message digest, 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.

13. Waiver Procedure: The Federal Information Security Management Act (FISMA) does not allow for waivers to a FIPS that is made mandatory by the Secretary of Commerce.

14. Where to Obtain Copies of the Standard: This publication is available electronically by accessing http://csrc.nist.gov/publications/. Other computer security publications are available at the same web site.

联邦信息
处理标准出版物 180-4

2015 年 8 月

发表

安全哈希标准
(SECURE HASH STANDARD)

联邦信息处理标准出版物(FIPS PUB)由美国国家标准与技术研究所根据 1996 年《信息技术管理改革法案》(公法 104-106)第 5131 节和 1987 年《计算机安全法》(公共法 100-235)的规定，经商务部张批准后发布。

1. **标准名称：**安全哈希标准 (*Secure Hash Standard*, SHA) (FIPS PUB 180-4)

2. **标准类别：**计算机安全标准(*Computer Security Standard*)、密码学(*Cryptography*)。

3. **说明：**本标准规定了用于计算电子数据(消息(*message*))的压缩表示的安全哈希算法 SHA-1、SHA-256、SHA-384、SHA-512，SHA-512/224 和 SHA-512/256。当任何长度小于2^{64}位(对于 SHA-1、SHA-224 和 SHA-256)或小于2^{128}(对于 SHA-384、SHA-512、SHA-512/224 和 SHA-512/256)的消息被输入到哈希算法(*hash algorithm*)时，结果是称为消息摘要(*message digest*)的输出。消息摘要的长度范围从 160 到 512 位，具体取决于算法。安全哈希算法通常与其它加密算法(*cryptographic algorithms*)一起使用，例如数字签名算法(*digital signature algorithms*)和密钥哈希消息认证码(*keyed-hash message authentication codes*)，用于生成随机数(比特(*bits*))。

本标准中规定的哈希算法被称为安全算法(*secure algorithms*)，因为对于给定算法，在计算上不可能找到与给定消息摘要相对应的消息，或者找到产生相同消息摘要的两个不同消息。对消息的任何更改都很有可能导致不同的消息摘要。当安全哈希算法与数字签名算法或密钥哈希消息认证算法一起使用时，这将导致验证失败。

本标准取代 FIPS 180-3 [FIPS 180-3]

4. **批准机关：**商务部长。

5. **维护机构：**美国商务部、国家标准与技术研究所(NIST)、信息技术实验室(ITL)。

6. **适用性**：本标准适用于所有联邦部门和机构，以保护不受《美国法典》第 10 篇第 2315 节(10 USC 2315)约束且不在《美国法典》第 40 篇第 11103(a)(1)(40 USC 11103(a)(1))节定义的国家安全体系内的敏感非机密信息。本标准或联邦信息处理标准(FIPS)202 必须在联邦应用需要安全哈希算法的任何地方实现，包括作为其它加密算法和协议中的组件。非联邦政府组织可采用和使用本标准。

7. **规范**：联邦信息处理标准(FIPS) 180-4，安全哈希标准(SHS) (附)。

8. **实现**：本文中指定的安全哈希算法可以在软件、固件、硬件或其任意组合中实现。只有经过 NIST 验证的算法实现才被视为符合本标准。有关验证程序信息，请访问：http://csrc.nist.gov/groups/STM/index.html

9. **实施时间表**：关于 FIPS 180-4 的测试和验证及其与 FIPS 140-2 的关系和指南可参见 FIPS PUB 140-2 实施指南 IG 1.10 和密码模块验证计划：http://csrc.nist.gov/groups/STM/cmvp/index.html.

10. **专利**：本标准中的安全哈希算法的实现可能包含在美国或外国专利中。

11. **出口管制**：某些密码设备及其相关技术数据受联邦出口管制。实施本标准的加密模块及其相关技术数据的出口必须符合这些联邦法规，并获得美国商务部出口管理的许可。有关出口法规的信息，请访问：http://www.bis.doc.gov/index.htm

12. **资格**：虽然本标准旨在规定生成消息摘要的一般安全要求，但符合本标准不能保证特定实现的安全性。各机构或部门的负责机构应确保总体实时提供可接收的安全水平。本标准将每五年审查一次，以评估其充分性。

13. **豁免程序**：《联邦信息安全管理法》(FISMA)不允许商务部长强制要求的 FIPS 豁免。

14. **从何处获取标准副本**：本出版物可通过访问 http://csrc.nist.gov/publications/ 来寻找。其它计算机安全出版物可在同一网站上查阅。

联邦信息
处理标准出版物 180-4

安全哈希标准

(SECURE HASH STANDARD)
技术规格

目录

1. INTRODUCTION (简介) 1
2. DEFINITIONS (定义) 2
   2.1. GLOSSARY OF TERMS AND ACRONYMS (术语表和缩略语) 2
3. NOTATION AND CONVENTIONS (符号和惯例) 4
   3.1. BIT STRING AND INTEGERS (位串和整数) 4
   3.2. OPERATIONS ON WORDS (字的操作) 5
4. FUNCTIONS AND CONSTANTS (函数和常量) 6
   4.1. FUNCTIONS (函数) 6
   4.2. CONSTANTS (常量) 7
5. PREPROCESSING (预处理) 8
   5.1. PADDING THE MESSAGE (填充消息) 8
   5.2. PARSING THE MESSAGE (分解消息) 9
   5.3. SETTING THE INITIAL HASH VALUE ($H0$) (设置初始哈希值) 9
6. SECURE HASH ALGORITHMS (安全哈希算法) 12
   6.1. SHA-1 12
   6.2. SHA-256 15
   6.3. SHA-224 17
   6.4. SHA-512 17
   6.5. SHA-384 19
   6.6. SHA-512/224 19
   6.7. SHA-512/256 19
7. TRUNCATION OF A MESSAGE DIGEST (消息摘要的截断) 20

APPENDIX A: ADDITIONAL INFORMATION 21

附录 A：附加信息 22

APPENDIX B: REFERENCES 23

ERRATUM 24

1. INTRODUCTION (简介)

本标准规定了安全哈希算法 SHA-1、SHA-224、SHA-256、SHA-384、SHA-512、SHA-512/224 和 SHA512/556。所有算法都是迭代的单项哈希函数，可以处理消息以生成称为消息摘要的压缩表示。这些算法能够确定消息的完整性：对详细的任何更改都很有可能导致不同的消息摘要。此属性在生成和验证数字签名和消息验证码以及生成随机数或比特时非常有用。

每种算法可以分为两个阶段进行描述：预处理 (*preprocessing*) 和哈希计算 (*hash computation*) 。预处理包括消息填充、将填充的消息解析为m位的块，以及设置哈希计算中的初始化值。哈希计算从填充的消息生成消息调度 (*message schedule*) ，并使用该调度以及函数 (*functions*) 、常量 (*constants*) 和字 (*word*) 操作来迭代生成一系列哈希值 (*hash values*) 。哈希计算生成的最终哈希值用于确定消息摘要。

这些算法在为被哈希的数据提供的安全强度方面上差异巨大。当这些哈希函数中的每一个与其它加密算法 (列如数字签名算法和密钥哈希消息认证码) 一起使用时，这些哈希功能和整个系统的安全强度可以在[SP 800-57]和[SP 800-107]中找到。

此外，算法在哈希或消息摘要大小期间使用的数据块和字的大小方面有所不同。图 1 展示了这些哈希算法的基本属性。

Algorithm *算法*	Message Size 消息大小(bits)	Block Size 块大小(bits)	Word Size 字大小(bits)	Message Digest Size 消息摘要大小(bits)
SHA-1	$< 2^{64}$	512	32	160
SHA-224	$< 2^{64}$	512	32	224
SHA-256	$< 2^{64}$	512	32	256
SHA-384	$< 2^{128}$	1024	64	384
SHA-512	$< 2^{128}$	1024	64	512
SHA-512/224	$< 2^{128}$	1024	64	224
SHA-512/256	$< 2^{128}$	1024	64	256

图 1 安全哈希算法属性

2. DEFINITIONS (定义)

2.1. Glossary of Terms and Acronyms (术语表和缩略语)

Bit	一位二进制数字，拥有两个值。0 和 1。
$Byte$	一字节由 8 位二进制组成。
$FIPS$	联邦信息处理标准。
$NIST$	美国国家标准与技术研究所。
SHA	安全哈希算法。
SP	特别出版物。
$Word$	一组 32 bits (4 bytes)或 64 bits (8 bytes)，这取决于安全哈希算法。

2.2. Algorithm Parameters, Symbols, and Terms (算法参数、符号和术语)

2.2.1 Parameters (参数)

以下参数用于本标准中的安全哈希算法规范：

$a, b, c, \ldots, h$	*Working variables* ，即哈希值$H^{(i)}$计算中使用得w-bit字。
$H^{(i)}$	第i个哈希值。$H^{(0)}$是初始哈希散列值；$H^{(N)}$是最终散列值，用于确定消息摘要。
$H_j^{(i)}$	第i个哈希值的第j个字，其中$H_0^{(i)}$是第i个哈希值的最左边的字。
K_t	用于哈希计算迭代t的常量值。
k	填充步骤期间附加到消息的零数。
ℓ	消息(M)长度，以位为单位。
m	消息块$(M^{(i)})$的位数 。
M	要进行哈希计算的消息。
$M^{(i)}$	第i个消息块，大小为$m\ bits$。
$M_j^{(i)}$	第i个消息块的第j个字，其中$M_0^{(i)}$是第i个消息块的最左边的字。

n 操作字时要循环位移或位移的位数。

N 填充消息中的块数。。

T 哈希计算中使用的临时$w\text{-}bit$字。

w 字中的位数。

W_t 消息调度的第t个$w\text{-}bit$字。

2.2.2 Symbols and Operations (符号和操作)

以下符号用于安全哈希算法规范；每个对$w\text{-}bit$的字进行操作。

$\wedge$ 按位与

$\vee$ 按位或

$\neg$ 按位补

$\oplus$ 按位异或

$\ll$ 左移操作，其中$x \ll n$是通过丢弃x最左边n位，然后在右边填充n个零来获得结果

$\gg$ 右移操作，其中$x \gg n$是通过丢弃x最右边n位，然后在左边填充n个零来获得结果

以下操作用于安全哈希算法规范：

$ROTL^n(x)$ 左循环 ***rotate left*** (循环左移 ***circular left shift***)，其中x是一个$w\text{-}bit$的字，n是整数 $(0 \le n < w)$，由 $ROTL^n(x) = (x \ll n) \vee (x \gg w - n)$定义。

$ROTR^n(x)$ 右循环 ***rotate right*** (循环右移 ***circular right shift***)，其中x是一个$w\text{-}bit$的字，n是整数 $(0 \le n < w)$，由$ROTR^n(x) = (x \gg n) \vee (x \ll w - n)$定义。

$SHR^n(x)$ 右移 $right\ shift$，其中x是一个$w\text{-}bit$的字，n是整数 $(0 \le n < w)$，由$SHR^n(x) = x \gg n$定义。

3. NOTATION AND CONVENTIONS (符号和惯例)

3.1. Bit String and Integers (位串和整数)

本文档将使用以下位串和整数相关的术语。

1. 十六进制(*hex digit*)数字是由集合{0,1,2,3,4,5,6,7,8,9, A, B, C, D, E, F}中的元素组成。一个十六进制数字表示4-bit的字符串。

   例如：$(7)_{16} = (0111)_2$　$(a)_{16} = (1010)_2$

2. 一个字是w-bit的字符串，可以表示为十六进制数字序列。为了将一个字转换为十六进制数字，每个4-bit字符串都被转换为其十六进制的等价符号。

   例如：　1010 0001 0000 0011 1111 1110 0010 0011
   a　1　0　3　f　e　2　3
   $A103FE23$

   在本规范中，在表示32-bit和64-bit字时使用大端序的约定，因此在每个字中，最高有效位存储在最左边的位位置。

3. 一个整数可以标识为一个字或一对字。消息长度的字表示ℓ，第5.1节的填充技术需要以位为单位。

   0和$2^{23} - 1$之间的整数可以表示为32-bit的字。整数的最低有效四位由字表示的最右十六进制数表示。

   例如：整数$291 = 2^8 + 2^5 + 2^1 + 2^0 = 256 + 32 + 2 + 1$由十六进制数字$(00000123)_{16}$表示。

   对于0到$2^{64} - 1$之间的整数(包括0和$2^{64} - 1$)也是如此，该整数可以表示为 64 位字。

   如果Z是整数，则$0 \le Z \le 2\text{^}64$，则$Z = 2^{32}X + Y$，其中$0 \le X < 2^{32}$且$0 \le Y < 2^{32}$。由于X和Y可以分别表示为32-bit的字X和Y，所以整数X可以表示为一对字(X, Y)。此属性用于 SHA-1、SHA-224 和 SHA-256。

   如果Z是整数，则$0 \le Z \le 2^{128}$，则$Z = 2^{64}X + Y$,其中$0 \le X < 2^{64}$且$0 \le Y \le 2^{64}$。由于X和Y可以分别表示为64-bit的字X和Y，所以整数Z可以表示为一对字(X, Y)。此属性用于 SHA-384、SHA-512、SHA-512/224 和 SHA-512/256。

4. 对于安全散列算法，消息块的大小(m-$bits$)取决于算法。
   a) 对于 SHA-1、SHA-224 和 SHA-256，每个消息块具有512 $bits$，表示为 16 个 32 位字的序列。
   b) 对于 SHA-348、SHA-512、SHA-512/224 和 SHA-512/256，每个消息块具

有1024 $bits$，表示为16个64-bit的字序列。

3.2. Operations on Words (字的操作)

以下操作适用于所有五种安全哈希算法中的w-bit字。SHA-1、SHA-224 和 SHA-256 对32-bit的字($w = 32$)进行操作，SHA-384、SHA-512、SHA-512/224 和 SHA-512/256 对64-bit 的字($w = 64$)进行操作。

1. 按位逻辑字操作：$\wedge$,$\vee$,$\oplus$ 和$\neg$ (参见 2.2.2)
2. 模2^w的加法：

   操作$x + y$的定义如下：字x和y表示整数X和Y，其中$0 \le X < 2^w$并且$0 \le Y \le 2^w$。对于正整数U和V，设$U \ mod \ V$为U除以V的余数。计算：

   $$Z = (X + Y) \ mod \ 2^w$$

   然后$0 \le Z \le 2^w$，将整数Z转换为字，Z，并定义$Z = x + y$。
3. 右移操作$SHR^n(x)$，其中x是w-bit 的字，n的范围是$0 \le n < w$，定义为：

   $$SHR^n(x) = x \gg n$$

   此操作用于 SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 和 SHA-512/256 算法。
4. 右循环(循环右移)操作$ROTR^n(x)$，其中x是一个w-bit 的字，n的范围是$0 \le n < w$，定义如下：

   $$ROTR^n(x) = (x \gg n) \vee (x \ll w - n)$$

   此操作用于 SHA-224, SHA-256, SHA-384, SHA-512, SHA- 512/224 和 SHA-512/256 算法。
5. 左循环(循环左移)操作$ROTL^n(x)$，其中 x 是一个 w-bit 的字，n 的范围是 0≤n<w，定义如下：

   $$ROTL^n(x) = (x \ll n) \vee (x \gg w - n)$$

   此操作仅在 SHA-1 算法中使用。
6. 注意以下等价关系，其中w在每个关系中是固定的：

   $$ROTL^n(x) \approx ROTR^{w-n}(x)$$
   $$ROTR^n(x) = ROTL^{w-n}(x)$$

4. FUNCTIONS AND CONSTANTS (函数和常量)

4.1. Functions (函数)

本节定义了每个算法使用的函数。尽管 SHA- 224, SHA-256, SHA-384,SHA-512, SHA-512/224 和 SHA-512/256 算法都使用类似的功能，但由于这些功能的输入输出是不同大小的字，因此它们的描述被分成了 SHA-224 和 SHA-256 (参见 4.1.2)。以及 SHA-358、SHA-512、SHA-512/334 和 SHA-512/256(第 4.1.3 节)。每个算法包括 $Ch(x,y,z)$和$Maj(x,y,z)$函数，异或运算(⊕)。在这些函数中，可以用 OR 操作代替(∨)并产生相同的结果。

4.1.1 SHA-1 函数

SHA-1 使用一系列逻辑函数$f_0, f_1, \ldots, f_{79}$。每个函数f_t $(0 \le t \le 79)$，对x、y和z这三个32-bit 的字进行运算并产生一个32-bit 的字作为输出。函数$f_t(x,y,z)$定义如下：

$$f_t(x,y,z) = \begin{cases} Ch(x,y,z)=(x \wedge y) \oplus (\neg x \wedge z) & 0 \le t \le 19 \\ Parity(x,y,z)=x \oplus y \oplus z & 20 \le t \le 39 \\ Maj(x,y,z)=(x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) & 40 \le t \le 59 \\ Parity(x,y,z)=x \oplus y \oplus z & 60 \le t \le 79. \end{cases} \quad (4.1)$$

4.1.2 SHA-224 和 SHA-256 函数

SHA-224 和 SHA-256 都使用六个逻辑函数，其中每个函数对32-bit 的字进行操作，这些字表示为x、y和z。每个函数的结果都是一个新的32-bit 的字。

$$Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z) \quad (4.2)$$

$$Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \quad (4.3)$$

$$\sum_0^{\{256\}}(x) = ROTR^{2}(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x) \quad (4.4)$$

$$\sum_1^{\{256\}}(x) = ROTR^{6}(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x) \quad (4.5)$$

$$\sigma_0^{\{256\}}(x) = ROTR^{7}(x) \oplus ROTR^{18}(x) \oplus SHR^{3}(x) \quad (4.6)$$

$$\sigma_1^{\{256\}}(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x) \quad (4.7)$$

4.1.3 SHA-384, SHA-512,SHA-512/224 和 SHA-512/256 函数

SHA-384, SHA-512, SHA-512/224 和 SHA-512/256 使用六个逻辑函数，其中每个函数对64-bit的字进行运算，这些字表示为x、y和z。每个函数的结果都是一个新的64-bit的字。

$$Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z) \quad (4.8)$$

$$Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \quad (4.9)$$

$$\sum_0^{\{512\}}(x) = ROTR^{28}(x) \oplus ROTR^{34}(x) \oplus ROTR^{39}(x) \quad (4.10)$$

$$\sum_1^{\{512\}}(x) = ROTR^{14}(x) \oplus ROTR^{18}(x) \oplus ROTR^{41}(x) \quad (4.11)$$

$$\sigma_0^{\{512\}}(x) = ROTR^{1}(x) \oplus ROTR^{8}(x) \oplus SHR^{7}(x) \quad (4.12)$$

$$\sigma_1^{\{512\}}(x) = ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^{6}(x) \quad (4.13)$$

4.2. Constants (常量)

4.2.1 SHA-1 常量

SHA-1 使用 80 个恒定的32-bit 字序列，$K_0, K_1, \ldots, K_{79}$。由下列式给出：

$$K_t = \begin{cases} \texttt{5a827999} & 0 \le t \le 19 \\ \texttt{6ed9eba1} & 20 \le t \le 39 \\ \texttt{8f1bbcdc} & 40 \le t \le 59 \\ \texttt{ca62c1d6} & 60 \le t \le 79 \end{cases} \quad (4.14)$$

4.2.2 SHA-224 和 SHA-256 常量

SHA-224 和 SHA-256 使用 64 个恒定32-*bit*字的相同序列，$K_0^{\{256\}}, K_1^{\{256\}}, \ldots, K_{63}^{\{256\}}$。这些字表示前 64 个素数的立方根分数部分的前 32 位。在十六进制中，这些常量字是：(从左到右)

```
428A2F98  71374491  B5C0FBCF  E9B5DBA5  3956C25B  59F111F1  923F82A4  AB1C5ED5
D807AA98  12835B01  243185BE  550C7DC3  72BE5D74  80DEB1FE  9BDC06A7  C19BF174
E49B69C1  EFBE4786  0FC19DC6  240CA1CC  2DE92C6F  4A7484AA  5CB0A9DC  76F988DA
983E5152  A831C66D  B00327C8  BF597FC7  C6E00BF3  D5A79147  06CA6351  14292967
27B70A85  2E1B2138  4D2C6DFC  53380D13  650A7354  766A0ABB  81C2C92E  92722C85
A2BFE8A1  A81A664B  C24B8B70  C76C51A3  D192E819  D6990624  F40E3585  106AA070
19A4C116  1E376C08  2748774C  34B0BCB5  391C0CB3  4ED8AA4A  5B9CCA4F  682E6FF3
748F82EE  78A5636F  84C87814  8CC70208  90BEFFFA  A4506CEB  BEF9A3F7  C67178F2
```

4.2.3 SHA-384, SHA-512, SHA-512/224 和 SHA-512/256 常量

SHA-384, SHA-512, SHA-512/224 和 SHA-512/256 使用 80 个恒定64-*bit*的字序列，$K_0^{\{512\}}, K_1^{\{512\}}, \ldots, K_{79}^{\{512\}}$。这些字表示前 80 个素数的立方根分数部分的前 64 位。在十六进制中，这些常量字是：(从左到右)

```
428A2F98D728AE22    7137449123EF65CD    B5C0FBCFEC4D3B2F    E9B5DBA58189DBBC
3956C25BF348B538    59F111F1B605D019    923F82A4AF194F9B    AB1C5ED5DA6D8118
D807AA98A3030242    12835B0145706FBE    243185BE4EE4B28C    550C7DC3D5FFB4E2
72BE5D74F27B896F    80DEB1FE3B1696B1    9BDC06A725C71235    C19BF174CF692694
E49B69C19EF14AD2    EFBE4786384F25E3    0FC19DC68B8CD5B5    240CA1CC77AC9C65
2DE92C6F592B0275    4A7484AA6EA6E483    5CB0A9DCBD41FBD4    76F988DA831153B5
983E5152EE66DFAB    A831C66D2DB43210    B00327C898FB213F    BF597FC7BEEF0EE4
C6E00BF33DA88FC2    D5A79147930AA725    06CA6351E003826F    142929670A0E6E70
27B70A8546D22FFC    2E1B21385C26C926    4D2C6DFC5AC42AED    53380D139D95B3DF
650A73548BAF63DE    766A0ABB3C77B2A8    81C2C92E47EDAEE6    92722C851482353B
```

A2BFE8A14CF10364	A81A664BBC423001	C24B8B70D0F89791	C76C51A30654BE30
D192E819D6EF5218	D69906245565A910	F40E35855771202A	106AA07032BBD1B8
19A4C116B8D2D0C8	1E376C085141AB53	2748774CDF8EEB99	34B0BCB5E19B48A8
391C0CB3C5C95A63	4ED8AA4AE3418ACB	5B9CCA4F7763E373	682E6FF3D6B2B8A3
748F82EE5DEFB2FC	78A5636F43172F60	84C87814A1F0AB72	8CC702081A6439EC
90BEFFFA23631E28	A4506CEBDE82BDE9	BEF9A3F7B2C67915	C67178F2E372532B
CA273ECEEA26619C	D186B8C721C0C207	EADA7DD6CDE0EB1E	F57D4F7FEE6ED178
06F067AA72176FBA	0A637DC5A2C898A6	113F9804BEF90DAE	1B710B35131C471B
28DB77F523047D84	32CAAB7B40C72493	3C9EBE0A15C9BEBC	431D67C49C100D4C
4CC5D4BECB3E42B6	597F299CFC657E2A	5FCB6FAB3AD6FAEC	6C44198C4A475817

5. PREPROCESSING (预处理)

预处理包括三个步骤：填充消息(M) (第 5.1 节)、将消息解析为消息块(第 5.2 节)和设置初始哈希值$H^{(0)}$(第 5.3 节)

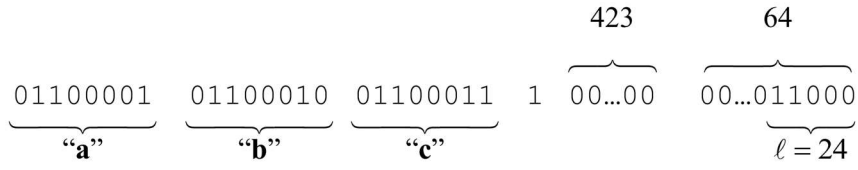
5.1. Padding the Message (填充消息)

此填充的目的是确保填充得消息是 512 或 1024 位的倍数，具体取决于算法。填充可以在消息开始哈希计算之前插入，也可以在处理将包含填充的块之前的哈希计算期间的任何其它时间插入。

5.1.1 SHA-1,SHA-224 和 SHA-256

假设消息的长度M为ℓ位。将1附加到消息的末尾，后跟k个0。其中，k是等式$(\ell + 1 + k \equiv 448\ mod\ 512)$最小的非负解。然后附加64-bet 的二进制块，该块等于使用二进制表示的数字ℓ。

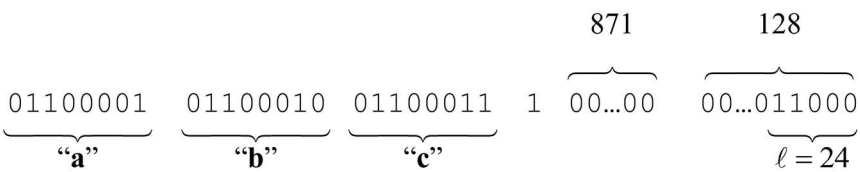
例如 (8-bit ASCII) 消息 “abc” 的长度为 $8 \times 3 = 24$，所以消息填充了 1 位，然后是$448 - (24 + 1) = 423$个 0。然后是消息长度(64-bit)，成为 512-bit 填充消息。



填充消息的长度限制是 512-bit 的倍数。

5.1.2 SHA-384、SHA-512、SHA-512/224 和 SHA-512/256

假设消息M的长度(以位为单位)为ℓ位。将 1 附加到消息末尾，后跟k个 0，其中k是等式$(\ell + 1 + k \equiv 896\ mod\ 1024)$的最小非负解。然后附加128-$bit$的二进制块，该块等于使用二进制表示的数字$\ell$。



填充消息的长度限制应是 1024-bit 的倍数。

5.2. Parsing the Message (分解消息)

消息及其填充必须分别为N个m位的块。

5.2.1 SHA-1, SHA-224 和 SHA-256

对于 SHA-1、SHA-224 和 SHA-256，消息及其填充被分解为 $M^{(1)}, M^{(2)}, \dots, M^{(N)}$ 由于输入块的 512 位可以表示为 16 个 32-bit 的字，因此消息块i的前 32-bit 表示为 $M_0^{(i)}$，后 32 位表示为$M_1^{(i)}$，以此类推，直到$M_{15}^{(i)}$。

5.2.2 SHA-384, SHA-512, SHA-512/224 和 SHA-512/256

对于 SHA-384, SHA-512, SHA-512/224 和 SHA-512/256，消息及其填充被分解为 N个 1024-bit 的块，$M^{(1)}, M^{(2)}, \dots, M^{(N)}$。由于输入块的 1024-bit 可以表示为 16 个 64-bit 的字，所以消息块i的前 64-bit 表示为$M_0^{(i)}$，后 64 位表示为$M_1^{(i)}$，以此类推，直到$M_{15}^{(i)}$

5.3. Setting the Initial Hash Value ($H^{(0)}$) (设置初始哈希值)

在针对每个安全哈希算法开始哈希计算之前，必须设置$H^{(0)}$。$H^{(0)}$中字的大小和数量取决于消息摘要的小大。

5.3.1 SHA-1

对于 SHA-1，初始哈希值$H^{(0)}$应包括以下五个 32-bit 字 (HEX)：

$$H_0^{(0)} = 67452301$$
$$H_1^{(0)} = EFCDAB89$$
$$H_2^{(0)} = 98BADCFE$$
$$H_3^{(0)} = 10325476$$
$$H_4^{(0)} = C3D2E1F0$$

5.3.2 SHA-224

对于 SHA-224，初始哈希值$H^{(0)}$应包括以下八个 32-bit 的字 (HEX)：

$$H_0^{(0)} = C1059ED8$$
$$H_1^{(0)} = 367CD507$$
$$H_2^{(0)} = 3070DD17$$
$$H_3^{(0)} = F70E5939$$
$$H_4^{(0)} = FFC00B31$$
$$H_5^{(0)} = 68581511$$
$$H_6^{(0)} = 64F98FA7$$
$$H_7^{(0)} = BEFA4FA4$$

5.3.3 SHA-256

对于 SHA-256，初始哈希值$H^{(0)}$应包括以下八个 32-bit 的字 (HEX)：

$$H_0^{(0)} = 6A09E667$$
$$H_1^{(0)} = BB67AE85$$
$$H_2^{(0)} = 3C6EF372$$
$$H_3^{(0)} = A54FF53A$$
$$H_4^{(0)} = 510E527F$$
$$H_5^{(0)} = 9B05688C$$
$$H_6^{(0)} = 1F83D9AB$$
$$H_7^{(0)} = 5BE0CD19$$

这些字是通过取前八个素数的平方根的分数部分的前 32 位得到的。

5.3.4 SHA-384

对于 SHA-384 哈希值$H^{(0)}$应包括以下八个 64it 的字 (HEX)：

$$H_0^{(0)} = \texttt{cbbb9d5dc1059ed8}$$
$$H_1^{(0)} = \texttt{629a292a367cd507}$$
$$H_2^{(0)} = \texttt{9159015a3070dd17}$$
$$H_3^{(0)} = \texttt{152fecd8f70e5939}$$
$$H_4^{(0)} = \texttt{67332667ffc00b31}$$
$$H_5^{(0)} = \texttt{8eb44a8768581511}$$
$$H_6^{(0)} = \texttt{db0c2e0d64f98fa7}$$
$$H_7^{(0)} = \texttt{47b5481dbefa4fa4}$$

这些字是通过取第九到第十六个素数平方根小数部分的前 64 位得到的。

5.3.5 SHA-512

对于 SHA-512 哈希值$H^{(0)}$应包括以下八个 64it 的字 (HEX)：

$$H_0^{(0)} = \texttt{6a09e667f3bcc908}$$
$$H_1^{(0)} = \texttt{bb67ae8584caa73b}$$
$$H_2^{(0)} = \texttt{3c6ef372fe94f82b}$$
$$H_3^{(0)} = \texttt{a54ff53a5f1d36f1}$$
$$H_4^{(0)} = \texttt{510e527fade682d1}$$
$$H_5^{(0)} = \texttt{9b05688c2b3e6c1f}$$
$$H_6^{(0)} = \texttt{1f83d9abfb41bd6b}$$
$$H_7^{(0)} = \texttt{5be0cd19137e2179}$$

这些字是通过取前八个素数平方根的分数部分的前 64 位得到的。

5.3.6 SHA-512/t

SHA512/t 是基于 SHA512 的 t 位哈希函数的总称，其输出被截断为 t 位。每个哈希函数都需要不同的初始哈希值。本节提供了确定给定值t的 SHA-512/t 初始值的程序。

对于 SHA-512/t， t是任何没有前导零的正整数，因此$t < 512$且t不是384。

对于给定的t值，SHA-512/t 的初始哈希值应由下面的 SHA-512/t IV 生成函数生成。

SHA-512/t IV Generation Function

(begin:)

Denote $H^{(0)}$ to be the initial hash value of SHA-512 as specified in Section 5.3.5 above.

表示$H^{(0)}$为上述5.3.5节中规定的SHA-512的初始值。

Denote $H^{(0)'}$ to be the initial hash value computed below.

表示$H^{(0)}$为下面计算的初始散列值/。

$H^{(0)}$ is the IV for SHA-512/t

$H^{(0)}$是SHA-512/t的IV

For $i = 0$ to 7

{

$H_i^{(0)'} = H_i^{(0)} \oplus$ a5a5a5a5a5a5a5a5(in hex).

}

$H^{(0)}$ = SHA-512 ("SHA-512/t") using $H^{(0)'}$ as the IV, where t is the specific truncation Value.

使用$H^{(0)}$=SHA-512("SHA-512/t")作为IV，其中t是特定截断值

(end.)

SHA-512/224（t＝224）和 SHA-512/556（t＝256）是经批准的哈希算法。根据需要，将来可能会在[SP 800-107]中指定具有不同 t 值的其他 SHA-512/t 哈希算法。以下是 SHA-512/224 和 SHA-512/556 的 IV。

5.3.6.2 SHA-512/256

For SHA-512/256, the initial hash value, $H^{(0)}$, shall consist of the following eight 64-bit words, in hex:

$$H_0^{(0)} = 22312194FC2BF72C$$
$$H_1^{(0)} = 9F555FA3C84C64C2$$
$$H_2^{(0)} = 2393B86B6F53B151$$
$$H_3^{(0)} = 963877195940EABD$$
$$H_4^{(0)} = 96283EE2A88EFFE3$$
$$H_5^{(0)} = BE5E1E2553863992$$
$$H_6^{(0)} = 2B0199FC2C85B8AA$$
$$H_7^{(0)} = 0EB72DDC81C52CA2$$

These words were obtained by executing the *SHA-512/t IV Generation Function* with $t = 256$.

6. SECURE HASH ALGORITHMS (安全哈希算法)

在下面的部分中，哈希算法没有按照大小的升序进行描述。SHA-256 在 SHA-224 之前进行了描述，因为 SHA-244 的规范与 SHA-256 相同，只是使用了不同的初始哈希值，并且 SHA-224 的最终的哈希值被截断为 224 位。SHA-512、SHA-384、SHA-512/224 和 SHA-512/256 也是如此，只是 SHA-512/224 的最终哈希值被截断至 224 位，SHA-512/256 为 256 位，SHA-384 为 384 位。

对于每个安全哈希算法，可能存在产生相同结果的替代计算方法，一个例子是第 6.1.3 节中描述的替代 SHA-1 计算。此类代替方法可按照本标准实施。

6.1. SHA-1

SHA-1 可用的哈希长度为ℓ位，其中$0 \le \ell < 2^{64}$。该算法使用 1) 80 个 32-bit 字的消息调度表，每个 32-bit 的五个工作变量，以及 5 个 32 位的哈希值。SHA-1 的最终结果是 160 位的消摘要。

消息调度表的字记为W_0、W_1、...、W_{79}。五个工作变量为a、b、c、d和e。其将保持初始哈希值$H^{(0)}$，由每个连续的中间哈希值(在每个消息块被处理之后)$H^{(i)}$代替，并以最终哈希值$H^{(N)}$结束。SHA-1 还使用一个临时字T。

6.1.2 SHA-1 Preprocessing (SHA-1 预处理)

1. 按照第 5.3.1 节规定设置初始哈希值$H^{(0)}$。

2. 按照第 5 节的规定填充和解析消息。

6.1.2 SHA-1 Hash Computation SHA-1 哈希计算

SHA-1 哈希计算分别使用第 4.1.1 节和第 4.2.1 节中定义的函数和常量。以2^{32}为，模进行加法(+)。

每个消息块$M^{(0)}$，$M^{(1)}$，...，$M^{(N)}$按顺序处理，使用以下步骤：

For i=1 to N:

{

1. Prepare the message schedule, $\{W_t\}$:

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ ROTL^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) & 16 \le t \le 79 \end{cases}$$

2. Initialize the five working variables, ***a***, ***b***, ***c***, ***d***, and ***e***, with the $(i\text{-}1)^{st}$ hash value:

$$a = H_0^{(i-1)}$$
$$b = H_1^{(i-1)}$$
$$c = H_2^{(i-1)}$$
$$d = H_3^{(i-1)}$$
$$e = H_4^{(i-1)}$$

3. For t=0 to 79:

{

$$T = ROTL^5(a) + f_t(b,c,d) + e + K_t + W_t$$

$$e = d$$

$$d = c$$

$$c = ROTL^{30}(b)$$

$$b = a$$

$$a = T$$

}

4. Compute the i^{th} intermediate hash value $H^{(i)}$:

$$H_0^{(i)} = a + H_0^{(i-1)}$$

$$H_1^{(i)} = b + H_1^{(i-1)}$$

$$H_2^{(i)} = c + H_2^{(i-1)}$$

$$H_3^{(i)} = d + H_3^{(i-1)}$$

$$H_4^{(i)} = e + H_4^{(i-1)}$$

}

After repeating steps one through four a total of N times (i.e., after processing $M^{(N)}$), the resulting 160-bit message digest of the message, M, is

$$H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)}$$

*在重复步骤 1 至 4 总共 N 此之后，即在处理M^N之后，得到的消息的 160-bit 消息摘要 M 为：

6.1.3 Alternate Method for Computing a SHA-1 Message Digest (计算 SHA-1 消息摘要的替代方法)

第 6.1.2 节中描述的 SHA-1 哈希计算方法假设消息调度W_0、W_1、...、W_{79}被实现为 80 个 32 位字的数组。从执行时间最小化的角度来看，这是有效的，因为第6.1.2节步骤(2)中W_{t-3}、...、W_{t-16}的地址很容易计算。

然而如果内存有限，另一种方法是将$\{W_t\}$视为一个循环队列，可用使用 16 个 32-bit 字的数组W_0、W_1、...、W_{15}来实现。本节所述的替代方法产生与第 6.1.2 节所述 SHA-1 计算方法相同的信息摘要。尽管该替代方法节省了 64 个32-bit 的存储字，由于步骤(3)中$\{W_t\}$的地址计算的复杂性增加，可能会延长执行时间。

对于这个备用 SHA-1 方法，让$MASK = 0000000F(\text{HEX})$。如第 6.1.1 节所述，以 232 为模进行加法计算。假设已执行第 6.1.1 条所述的预处理，$M^{(i)}$的处理如下：

For i=1 to N:

{

1. For t=0 to 15:

   {

   $$W_t = M_t^{(i)}$$

   }

2. Initialize the five working variables, ***a***, ***b***, ***c***, ***d***, and ***e***, with the (i-1)st hash value:

   $$a = H_0^{(i-1)}$$
   $$b = H_1^{(i-1)}$$
   $$c = H_2^{(i-1)}$$
   $$d = H_3^{(i-1)}$$
   $$e = H_4^{(i-1)}$$

3. For t=0 to 79:

   {

   $$s = t \wedge MASK$$

   If $t \geq 16$ then

   {

   $$W_s = ROTL^1(W_{(s+13)\wedge MASK} \oplus W_{(s+8)\wedge MASK} \oplus W_{(s+2)\wedge MASK} \oplus W_s)$$

   }

   $$T = ROTL^5(a) + f_t(b,c,d) + e + K_t + W_s$$
   $$e = d$$
   $$d = c$$
   $$c = ROTL^{30}(b)$$
   $$b = a$$
   $$a = T$$

   }

4. Compute the i^{th} intermediate hash value $H^{(i)}$:

   $$H_0^{(i)} = a + H_0^{(i-1)}$$
   $$H_1^{(i)} = b + H_1^{(i-1)}$$
   $$H_2^{(i)} = c + H_2^{(i-1)}$$
   $$H_3^{(i)} = d + H_3^{(i-1)}$$
   $$H_4^{(i)} = e + H_4^{(i-1)}$$

}

After repeating steps one through four a total of N times (i.e., after processing $M^{(N)}$), the resulting 160-bit message digest of the message, M, is

$$H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)}$$

6.2. SHA-256

SHA-256 可用于需要哈希的消息长度为 ℓ 位的消息，其中$0 \le \ell < 2^{64}$。该算法使用 1) 64 个 32-bit 的消息调度。2) 8 个 32-bit 工作变量，以及 3) 8 个 64-bit 字的哈希值。SHA-256 的最终结果是 256 位消息摘要。

消息调度表的字被标记为W_0、W_1、… 、W_{63}。八个工作变量被标记为a、b、c、d、e、f、g和h。哈希值的字被标记为$H_0^{(i)}$、$H_1^{(i)}$、…、$H_7^{(i)}$。将保持哈希值。$H^{(0)}$由每个连续的中间哈希值替换。

(在处理每个消息块之后)，$H^{(i)}$，并以最终哈希值$H^{(N)}$结束。SHA-256 还使用两个临时字T_1和T_2。

6.2.1 SHA-256 Preprocessing (SHA-256 预处理)

1. 按照第 5.3.3 节的规定设置初始哈希值$H^{(0)}$。
2. 按照第 5 节的规定填充和分解消息。

6.2.2 SHA-256 Hash Computation (SHA-256 哈希计算)

SHA-256 哈希计算分别使用 4.1.2 节和 4.2.2 节中定义的函数和常量。以2^{32}为模进行加法(+)。

每个消息块$M^{(1)}$、$M^{(2)}$、…、$M^{(N)}$按顺序处理，使用以下步骤：

For i = 1 to N

{

1. 准备消息序列，$\{M_t\}$：

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ \sigma_1^{\{256\}}(W_{t-2}) + W_{t-7} + \sigma_0^{\{256\}}(W_{t-15}) + W_{t-16} & 16 \le t \le 63 \end{cases}$$

2. 使用第$(i-1)^{st}$个哈希值初始化八个工作变量a、b、c、d、e、f、g和h：

$$a = H_0^{(i-1)}$$
$$b = H_1^{(i-1)}$$
$$c = H_2^{(i-1)}$$
$$d = H_3^{(i-1)}$$
$$e = H_4^{(i-1)}$$
$$f = H_5^{(i-1)}$$
$$g = H_6^{(i-1)}$$
$$h = H_7^{(i-1)}$$

3. **For** t = 0 to 63

{

$$T_1 = h + \sum_1^{\{256\}}(e) + Ch(e,f,g) + K_t^{\{256\}} + W_t$$

$$T_t = \sum_0^{\{256\}}(a) + Maj(a,b,c)$$

$$h = g$$
$$g = f$$
$$f = e$$
$$e = d + T_1$$
$$d = c$$
$$c = b$$
$$b = a$$
$$a = T_1 + T_2$$

}

4. 计算第i^{th}个中间哈希值$H^{(i)}$：

$$H_0^{(i)} = a + H_0^{(i)}$$
$$H_1^{(i)} = b + H_1^{(i)}$$
$$H_2^{(i)} = c + H_2^{(i)}$$
$$H_3^{(i)} = d + H_3^{(i)}$$
$$H_4^{(i)} = e + H_4^{(i)}$$
$$H_5^{(i)} = f + H_5^{(i)}$$
$$H_6^{(i)} = g + H_6^{(i)}$$
$$H\text{^}((i))\_7 = h + H_7^{(i)}$$

}

再重复步骤 1 至 4 总共 N 次之后(即，在处理$M^{(N)}$之后)得到的消息的 256-bit 消息摘要M为：

$$H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)} \| H_5^{(N)} \| H_6^{(N)} \| H_7^{(N)}$$

6.3. SHA-224

SHA-224 可用于哈希长度为ℓ位的消息(M)，其中 $0 \le \ell < 2^{64}$。该函数的定义方式与 SHA-256(第 6.2 节)完全相同，但有以下两个例外：

1. 按照第 5.3.2 节的规定设置初始哈希值$H^{(0)}$。

2. 通过将最终哈希值$H^{(N)}$截断到其最左边的224-bits 来获得 224 位信息摘要。

$$H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)} \| H_5^{(N)} \| H_6^{(N)} \| H_7^{(N)}$$

6.4. SHA-512

SHA-512 可用于需要哈希的消息长度为 ℓ 位的消息，其中$0 \le \ell < 2^{128}$。该算法使用 1) 80 个 64-bit 的消息调度。2) 8 个 64-bit 工作变量，以及 3) 8 个 64-bit 字的哈希值。SHA-512 的最终结果是 512 位消息摘要。

消息调度表的字被标记为W_0、W_1、… 、W_{63}。八个工作变量被标记为a、b、c、d、e、f、g和h。哈希值的字被标记为$H_0^{(i)}$、$H_1^{(i)}$、...、$H_7^{(i)}$。将保持哈希值。$H^{(0)}$由每个连续的中间哈希值替换。

(在处理每个消息块之后)，$H^{(i)}$，并以最终哈希值$H^{(N)}$结束。SHA-512 还使用两个临时字T_1和T_2。

6.4.1 SHA-512 Preprocessing (SHA-512 预处理)

1. 按照第 5.3.5 节的规定设置初始哈希值$H^{(0)}$。

2. 按照第五节的规定填充和分解消息。

6.4.2 SHA-512 Hash Computation (SHA-512 哈希计算)

SHA-512哈希计算分别使用4.1.3节和4.2.3节中定义的函数和常量。以2^{64}为模进行加法(+)。

每个消息块$M^{(1)}$、$M^{(2)}$、...、$M^{(N)}$按顺序处理，使用以下步骤：

For i=1 to N:

{

1. Prepare the message schedule, $\{W_t\}$:

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ \sigma_1^{\{512\}}(W_{t-2}) + W_{t-7} + \sigma_0^{\{512\}}(W_{t-15}) + W_{t-16} & 16 \le t \le 79 \end{cases}$$

2. Initialize the eight working variables, ***a***, ***b***, ***c***, ***d***, ***e***, ***f***, ***g***, and ***h***, with the (i-1)st hash value:

$$a = H_0^{(i-1)}$$
$$b = H_1^{(i-1)}$$
$$c = H_2^{(i-1)}$$
$$d = H_3^{(i-1)}$$
$$e = H_4^{(i-1)}$$
$$f = H_5^{(i-1)}$$
$$g = H_6^{(i-1)}$$
$$h = H_7^{(i-1)}$$

3. For t=0 to 79:

{

$$T_1 = h + \sum_1^{\{512\}}(e) + Ch(e,f,g) + K_t^{\{512\}} + W_t$$
$$T_2 = \sum_0^{\{512\}}(a) + Maj(a,b,c)$$
$$h = g$$
$$g = f$$
$$f = e$$
$$e = d + T_1$$
$$d = c$$
$$c = b$$
$$b = a$$
$$a = T_1 + T_2$$

}

4. Compute the i^{th} intermediate hash value $H^{(i)}$:

$$H_0^{(i)} = a + H_0^{(i-1)}$$
$$H_1^{(i)} = b + H_1^{(i-1)}$$
$$H_2^{(i)} = c + H_2^{(i-1)}$$
$$H_3^{(i)} = d + H_3^{(i-1)}$$
$$H_4^{(i)} = e + H_4^{(i-1)}$$
$$H_5^{(i)} = f + H_5^{(i-1)}$$
$$H_6^{(i)} = g + H_6^{(i-1)}$$
$$H_7^{(i)} = h + H_7^{(i-1)}$$

}

After repeating steps one through four a total of N times (i.e., after processing $M^{(N)}$), the resulting 512-bit message digest of the message, M, is

$$H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)} \| H_5^{(N)} \| H_6^{(N)} \| H_7^{(N)}$$

6.5. SHA-384

SHA-384 可用于哈希长度为ℓ位的消息(M)，其中 $0 \leq \ell < 2^{128}$。该函数的定义方式与 SHA-512(第 6.4 节)完全相同，但有以下两个例外：

1. 按照第 5.3.4 节的规定设置初始哈希值$H^{(0)}$。

2. 通过将最终哈希值$H^{(N)}$截断到其最左边的384-bits 来获得 384 位信息摘要。

$$H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)} \| H_5^{(N)}$$

6.6. SHA-512/224

SHA-512/224 可用于哈希长度为ℓ位的消息(M)，其中 $0 \leq \ell < 2^{128}$。该函数的定义方式与 SHA-512(第 6.4 节)完全相同，但有以下两个例外：

1. 按照第 5.3.6.1 节的规定设置初始哈希值$H^{(0)}$。

2. 通过将最终哈希值$H^{(N)}$截断到其最左边的224-bits 来获得 224 位信息摘要。

6.7. SHA-512/256

SHA-512/256 可用于哈希长度为ℓ位的消息(M)，其中 $0 \leq \ell < 2^{128}$。该函数的定义方式与 SHA-512(第 6.4 节)完全相同，但有以下两个例外：

1. 按照第 5.3.6.2 节的规定设置初始哈希值$H^{(0)}$。

2. 通过将最终哈希值$H^{(N)}$截断到其最左边的256-bits 来获得 256 位信息摘要。

7. TRUNCATION OF A MESSAGE DIGEST (消息摘要的截断)

Some application may require a hash function with a message digest length different than those provided by the hash functions in this Standard. In such cases, a truncated message digest may be used, whereby a hash function with a larger message digest length is applied to the data to be hashed, and the resulting message digest is truncated by selecting an appropriate number of the leftmost bits. For guidelines on choosing the length of the truncated message digest and information about its security implications for the cryptographic application that uses it, see SP 800-107 [SP 800-107].

某些应用程序可能需要消息摘要长度不同于本标准中散列函数所提供的消息摘要长度的散列函数。在这种情况下，可以使用截短的消息摘要，从而将具有更大消息摘要长度的哈希函数应用于要散列的数据，并且通过选择适当数量的最左边的比特来截短所得到的消息摘要。有关选择截短消息摘要长度的指南以及有关其对使用该摘要的加密应用程序的安全影响的信息，请参见 SP 800-107[SP 800-107]。

APPENDIX A: Additional Information

A.1 Security of the Secure Hash Algorithms

The security of the five hash algorithms, SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256 is discussed in [SP 800-107].

A.2 Implementation Notes

Examples of SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 and SHA-512/256 are available at:

http://csrc.nist.gov/groups/ST/toolkit/examples.html

A.3 Object Identifiers

Object identifiers (OIDs) for the SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA- 512/224 and SHA-512/256 algorithms are posted at:

http://csrc.nist.gov/groups/ST/crypto_apps_infra/csor/algorithms.html

附录 A：附加信息

A.1 安全哈希算法的安全性

[SP800-107]中讨论了五种哈希算法 SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 和 SHA-512/256 的安全性。

A.2 实施说明

SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 和 SHA-512/256 的示例可用在如下网址找到:

http://csrc.nist.gov/groups/ST/toolkit/examples.html

A.3 对象标识符

SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA- 512/224 和 SHA-512/256 算法的对象标识符 Object identifiers (OIDs)可在如下网址找到:

http://csrc.nist.gov/groups/ST/crypto_apps_infra/csor/algorithms.html

APPENDIX B: REFERENCES

附录 B：参考文献

[FIPS 180-3] NIST, Federal Information Processing Standards Publication 180-3, *Secure Hash Standards (SHS)*, October 2008.

[SP 800-57] NIST Special Publication (SP) 800-57, Part 1, *Recommendation for Key Management: General*, (Draft) May 2011.

[SP 800-107] NIST Special Publication (SP) 800-107, *Recommendation for Applications Using Approved Hash Algorithms*, (Revised), (Draft) September 2011.

ERRATUM

The following change has been incorporated into FIPS 180-4, as of the date indicated in the table.

DATE	TYPE	CHANGE	PAGE NUMBER
5/9/2014	Editorial	Change "$t < 79$" to "$t \leq 79$"	Page 10, Section 4.1.1, Line 1