

对哈希算法 SHA-1 的分析和改进

林雅榕, 侯整风

(合肥工业大学 计算机科学与技术系, 安徽 合肥 230009)

摘要:研究了哈希算法的相关问题,对常用哈希算法 SHA-1 从安全性和运算效率方面进行了较深入分析,并由此提出了对该算法的几点改进,使改进后的算法在安全性及运算效率方面较原算法均有所提高。同时还提出了一种安全散列值的计算方法。

关键词:哈希算法; SHA-1; 安全; 散列值

中图分类号: TP393.08

文献标识码: A

文章编号: 1005-3751(2006)03-0124-03

Analysis and Improvement to Algorithm of SHA-1

LIN Ya-rong, HOU Zheng-feng

(Dept. of Computer Science and Technology, Hefei University of Technology, Hefei 230009, China)

Abstract: Study Hash algorithm related issues. By making an overall analysis to the algorithm of SHA-1, several improving schemes are proposed to enhance the security and the efficiency of the algorithm. Additionally, a new method of generating secure message digest is proposed.

Key words: Hash algorithm; SHA-1; security; message digest

0 引言

数字签名作为一种重要的安全技术如今已被广泛地应用于网络信息交换领域,它的根本作用是保证网络中传输的数据的真实性和完整性。而在种类众多的数字签名技术中通常都会用到一类特殊的数学算法——哈希算法,它在数字签名技术中占有极其重要的地位。哈希算法也即散列算法,其作用是对任何不定长的比特串(称为消息)计算出一个定长的比特串(称为消息摘要或散列值)。目前常见的哈希算法有 MD5^[1]、SHA-1^[2]和 RIPEMD-160^[3],而国内更倾向于 MD5 和 SHA-1。就当前的情况来看,SHA-1 由于其安全强度及运算效率方面的优势目前已经成为使用最为广泛的哈希算法了。

1 SHA-1 算法概述

SHA-1 算法由美国国家标准技术研究院(NIST)与美国国家安全局(NSA)设计,并且被美国政府采纳,成为美国国家标准^[4]。事实上 SHA-1 目前是世界使用最为广泛的哈希算法,已经成为业界的事实标准。SHA-1 可以对长度不超过 2^{64} 比特的消息进行计算,产生 160 比特的消息摘要作为输出。该算法的处理流程大致分为 5 个步骤:

- 1)在待处理的消息后面添加一个 1 和若干(至少 1 个,最多 512 个)个 0,使消息的长度变成 512 的倍数减去 64。
- 2)在新得到的消息后面再添加一个 64 比特的二进制串,其值为消息的原始长度,此时消息的长度为 512 的倍数。
- 3)初始化缓存,这里的缓存为 5 个 32 比特的变量。
- 4)利用主循环每次处理一个 512 比特的分组。主循环共有 4 轮,每轮 20 次操作。
- 5)输出 160 比特的消息摘要作为运算结果。

2 SHA-1 的安全性及运算效率分析

就目前密码学研究的最新进展来看,MD5 的安全性已经受到质疑^[5],而 RIPEMD-160 的安全强度是最高的。但 RIPEMD-160 由于采用二个循环体,五轮模块压缩函数逻辑结构,虽然增强了安全性,却使运算速度大大降低了。相比较而言,SHA-1 目前仍然是一种安全可靠的算法,其产生的 160 bit 散列值比起 MD5 的 128 bit 散列值具有高得多的安全强度。如果实施生日攻击,对 MD5 只要进行 2^{64} 次的运算,而对 SHA-1 (包括 RIPEMD-160)就要进行 2^{80} 次运算。对于穷举攻击,MD5 和 SHA-1 的运算次数分别为 2^{128} 和 2^{160} 。就运算速度而言,MD5 稍快,SHA-1 与之相差不多。另外,虽然 SHA-1 的安全性目前也在不断受到挑战,但是这些新的发现只是稍稍加快了找到强无碰撞的速度,尽管如此,其巨大的计算量

收稿日期:2005-06-29

作者简介:林雅榕(1968—),女,福建福州人,硕士研究生,研究方向为计算机网络、网络安全;侯整风,博士,教授,硕士生导师。

对于目前的硬件水平而言仍然是没有意义的;而要想真正对安全构成威胁,必须找到弱无碰撞,而其计算量相对于找到强无碰撞而言更是高得多。因此,SHA-1 在目前以及今后的几年内仍然是安全的。但是这些挑战毕竟给 SHA-1 敲响了警钟,故有必要对该算法进行一些改进,以使其具有更好的可用性。

从安全性及运算效率的角度来说,一个好的哈希算法应具备以下特性:

- 1) 对任意长度的消息能够计算出一个定长的且唯一的消息摘要;
- 2) 由一个已知的摘要不能反推出产生该摘要的消息;
- 3) 要找到两个具有相同摘要的不同消息在计算上是不可行的,尽管理论上是存在的;
- 4) 能够抗弱冲突,也能抗强冲突;
- 5) 具有较快的运算速度。

正是针对上述特性,文中对 SHA-1 算法提出几点改进,同时提出一种安全散列值的计算方法。

3 对哈希算法 SHA-1 的几点改进

下面将从分组长度、散列值的计算方式、逻辑函数的表达式和压缩函数逻辑结构 4 个方面对 SHA-1 进行改进。

3.1 扩大分组长度

SHA-1 算法中将长度小于 2^{64} 的消息凑位后分成 L 个 512 bit 的分组,每次处理一个分组。算法首先将原来的 512 bit 即 16 个 32 bit 字,按一定规则扩充为 80 个 32 bit 字,用来满足总体逻辑结构设计的 4 个循环模块及每个模块 20 个步骤(共计 80 步操作)的需要。这里用 W_i 表示其中任意一个 32 bit 字,则在这 80 步逻辑函数运算操作中,每一步处理一个 W_i ,共计处理 80 个 W_i 。这种做法意味着在一个扩充成 80 个 32 bit 字的分组内部引入了大量的冗余和相关,当然这也是为了增加算法的安全性。由此,可以设想,对于数据量大的长消息,如果将分组长度由原来规定的 512 bit 即 16 个 32 bit 字扩大一倍,改为 1024 bit 即 32 个 32 bit 字,然后仍按 SHA-1 算法字 W_i 扩充算法扩充为 80 个 32 bit 字 X_i ,这样分组内部同样引入了很多的冗余和相关,且算法的运算速度却因而提高,又不致影响算法的安全性。

3.2 安全散列值计算

由于散列算法是公开的,恶意节点可能会截获处于传输过程中的消息,修改消息内容,然后重新计算散列值并替换原来的散列值,使消息的完整性遭到破坏。为此在计算散列值时可以采取在消息中增添一个随机的 512 bit 分组的方法,从而增加安全系数。该随机分组的内容可以是消息的序列号或时间戳(以微秒为单位)的某种变换,例如将序列号或者时间戳的值左循环移动 $5 + 2n$ 位;该随机分组可以添加到原消息的后面,也可插入到原消息的任意位置。关于随机分组的内容及插入位置通信双方最好事

先共享一个秘密的规则库。具体工作流程如图 1 所示。

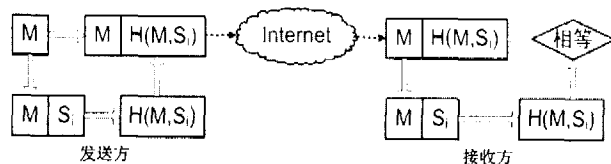


图1 安全散列值工作流程

* 发送方的工作:

- 1) 产生要发送的原始消息 M 。
- 2) 根据该原始消息的序列号或者时间戳产生随机分组 S_i 。
- 3) 将 S_i 添加到原始消息尾部或者事先约定的其它位置,对添加了随机分组的消息计算散列值 $H(M, S_i)$ 。
- 4) 将原始消息 M 与散列值 $H(M, S_i)$ 一同发送出去。

* 接收方的工作:

- 1) 从接收到的报文中分离出原始消息 M 与散列值 $H(M, S_i)$ 。
- 2) 同样根据序列号或者时间戳产生随机分组 S_i 。
- 3) 将 S_i 添加到原始消息尾部或者事先约定的其它位置,对添加了随机分组的消息计算散列值 $H(M, S_i)$ 。
- 4) 将计算出的散列值与接收到的散列值进行比较,若相等则表明消息没有被修改。由于秘密规则库、随机分组 S_i 的内容与插入位置都没有发送,所以攻击者无法根据截获的消息计算出正确的散列值,也就难以实施攻击。

可见按上述方法计算出的散列值不仅可以有效地保证消息的完整性,还可以起到序列号或时间戳的作用。由于这种类型的散列值增强了消息的安全性,可以称其为安全散列值。

3.3 变换原始逻辑函数

SHA-1 压缩函数逻辑结构有四轮循环模块,每一轮用一个逻辑函数。这里用 f_1, f_2, f_3, f_4 表示四轮循环对应的逻辑函数,用 AND, OR, NOT 和 XOR 分别表示逻辑运算符与、或、非和异或,则这 4 个函数的表达式如下:

$$f_1 = (B \text{ AND } C) \text{ OR } (\text{NOT } B \text{ AND } D)$$

$$f_2 = B \text{ XOR } C \text{ XOR } D$$

$$f_3 = (B \text{ AND } C) \text{ OR } (B \text{ AND } D) \text{ OR } (C \text{ AND } D)$$

$$f_4 = B \text{ XOR } C \text{ XOR } D$$

上述 4 个函数中由于 $f_2 = f_4$,所以 SHA-1 实际上只使用了 3 个逻辑函数。这样的话随着攻击者技术手段的不断完善,势必会给防御攻击带来一定的风险。解决办法:可以对 f_4 的函数表达式进行适当的变动,例如将 f_4 函数表达式改为 $(B \text{ AND } \text{NOT } C) \text{ XOR } D$ 或其他的函数表达式。这样改进不多,对其运算速度几乎没有影响,同时却相对提高了安全性。改进后的 4 个逻辑函数表达式如表 1 所示。

3.4 变换压缩函数逻辑结构

SHA-1 算法压缩函数逻辑结构,是以 512 bit 分组为单位进行处理的。主循环含有四轮循环模块,输入是当前分组 q 的 16 个字 X_i ,和由前一分组输出的 160 bit 缓存值

(在寄存器 A, B, C, D, E 中)。压缩函数逻辑结构对其进行四轮, 每轮 20 个步骤的运算, 每一步骤只处理一个字 W_t 。全部 L 个分组处理完毕后输出 160 bit 散列值。表 2 中列出了依照下式运算后头 5 个步骤寄存器的缓存值。

表 1 改进后的逻辑函数表达式

轮数	逻辑函数	函数表达式
第一轮 ($0 \leq t \leq 19$)	$f_1(B, C, D)$	(B AND C) OR (NOT B AND D)
第二轮 ($20 \leq t \leq 39$)	$f_2(B, C, D)$	B XOR C XOR D
第三轮 ($40 \leq t \leq 59$)	$f_3(B, C, D)$	(B AND C) OR (B AND D) OR (C AND D)
第四轮 ($60 \leq t \leq 79$)	$f_4(B, C, D)$	(B AND NOT C) XOR D

$A, B, C, D, E \leftarrow (E + f_t(B, C, D) + S^5(A) + W_t + K_t), A, S^{30}(B), C, D$

表 2 寄存器缓存值

步骤	寄存器缓存值				
0	A	B	C	D	E
1	E_1	A	B^1	C	D
2	D_1	E_1	A^1	B^1	C
3	C_1	D_1	E_1^1	A^1	B^1
4	B_1^1	C_1	D_1^1	E_1^1	A^1
5	A_1^1	B_1^1	C_1^1	D_1^1	E_1^1

上式中, W_t 分别是 t 步骤处理的字 W_t , $+$ 表示 $\text{mod}2^{32}$ 加法。 A, B, C, D, E 是处理 q 分组时寄存器的起始值。表 2 中各种符号代表值如下:

$$\begin{aligned} A^1 &= S^{30}(A) & B^1 &= S^{30}(B) \\ C_1 &= C + f_1(E_1, A^1, B^1) + S^5(D_1) + W_3 + k_1 \\ D_1 &= D + f_1(A, B^1, C) + S^5(E_1) + W_2 + k_1 \\ E_1 &= E + f_1(B, C, D) + S^5(A) + W_1 + k_1 \\ A_1^1 &= A^1 + f_1(C_1, D_1^1, E_1^1) + S^5(B_1^1) + W_5 + k_1 \\ B_1^1 &= B^1 + f_1(D_1, E_1^1, A^1) + S^5(C_1) + W_4 + k_1 \\ C_1^1 &= S^5(C_1) & D_1^1 &= S^5(D_1) & E_1^1 &= S^5(E_1) \end{aligned}$$

据此, 可以写出步骤 10, 15, 20 以及第二、三、四轮循环模块运算完毕后 5 个寄存器的缓存值。由此可见算法中上述规律性的存在是一个安全隐患。

SHA-1 算法中分组 q 经四轮逻辑运算产生的输出结果 $(A, B, C, D, E)_q$, 尚需和 5 个寄存器起始缓存值 CV_q 按 $\text{mod}2^{32}$ 相加后, 才产生分组 q 的最终输出 CV_{q+1} 。可表示成:

$$\begin{aligned} CV_0 &= CV_0 \\ CV_{q+1} &= \text{SUM}_{32}(CV_q, (A, B, C, D, E)_q) \end{aligned}$$

$CV_L = \text{散列值 MD}$

原 SHA-1 算法只是二个 32bit 字的输入, 产生一个 32bit 字的输出。如果先对每个寄存器进行一次模加法, 再进行 SHA-1 规定的其余的运算, 那么对抵抗强抗冲突将会更加有效, 更好地隐藏了摘要和明文之间的潜在规律。也即采用三个 32bit 字的输入产生一个 32bit 字的输出。

具体说, 将分组 q 寄存器的缓存起始值, 按顺序每两个进行 $\text{mod}2^{32}$ 加法运算后得出 $CV_q(AA, BB, CC, DD, EE)$, 再和分组 q 经四轮循环运算输出 $(A, B, C, D, E)_q$, 相应作 $\text{mod}2^{32}$ 加法运算后的输出, 得出最终的输出值 CV_{q+1} 。全部过程可表示为:

$$\begin{aligned} CV_0 &= CV_0 \\ CV_{q+1} &= \text{SUM}_{32}(CV_q(AA, BB, CC, DD, EE), (A, B, C, D, E)_q) \end{aligned}$$

$CV_L = \text{MD}$

$$\begin{aligned} AA &= B + C & BB &= C + D & CC &= D + E \\ DD &= E + A & EE &= A + B \end{aligned}$$

可见上述对压缩函数逻辑结构所做的变换在基本不增加计算量的同时, 却提高了抵抗强无碰撞的能力。

4 结束语

文中从安全性及运算效率方面对散列算法 SHA-1 作了深入分析, 并由此提出若干改进方法。改进后的算法在安全性及运算效率方面均较原算法有所提高, 因此具有更好的可用性。文中还提出了一种安全散列值的计算方法, 该方法产生的散列值不仅具有更高的安全性, 而且还具有序列号或时间戳的功能。

参考文献:

- [1] Rivest R. The MD5 Message - Digest Algorithm[S]. RFC 1321, 1992.
- [2] Eastlake D, Jones P. US Secure Hash Algorithm1 (SHA1) [S]. RFC 3174, 2001.
- [3] Dobbertin H, Bosselaers A, Preneel B. RIPMEMD - 160: A strengthened version of RIPMMD[Z]. Fast Software Encryption, 1996, LNCS 1039: 71 - 82.
- [4] National Institute of Standards and Technology. Secure hash standard. NISTFIPS PUB 180 - 1, Washington D C: Department of Commerce, NIST, 1995[EB/OL]. <http://csrc.nist.gov/cryptval/shs.html>. 1995.
- [5] 王小云, 张金清. MD5 报文摘要算法的各圈函数碰撞分析[J]. 计算机工程与科学, 1996, 18(2): 15 - 22.

(上接第 121 页)

机械工业出版社, 2004.

- [2] BUILDER COM. MVC 设计模式带来更好的软件结构和代码重用[DB/OL]. <http://www.zdnet.com.cn/developer/tech/story/0,2000081602,39098006,00.htm>, 2002 - 11.

- [3] The Apache Software Foundation: Struts[DB/OL]. <http://struts.apache.org/>, 2005 - 03.
- [4] OPENSYPHONY Quality Components[EB/OL]. <http://www.opensymphony.com/sitemesh/>, 2005 - 03.