

散列算法 MD5 和 SHA-1 的比较

王泽¹, 曹莉莎²

(1.华北电力大学 信息与网络管理中心,河北 保定 071000;2.保定学院 网络信息管理中心,河北 保定 071000)

摘要:MD5 和 SHA-1 是两个已知的广泛应用于信息安全的散列算法,均由 MD4 发展而来。详细介绍了它们的算法逻辑,并通过模拟实验、软件测试等方式对其在各个方面进行比较,最终得出结论:SHA-1 算法比 MD5 算法的安全性更高,但在同一硬件上,SHA-1 比 MD5 运行的要慢。

关键词:散列算法;SHA-1;MD5;消息认证;摘要

中图分类号:TN918.1 文献标识码:A 文章编号:1009-3044(2016)11-0246-02

Comparison between MD5 and SHA-1

WANG Ze¹, CAO Li-sha²

(1.Information & Network Management Center, North China Electric Power University, Baoding 071000, China;2.Network Information Management Center, Baoding University, Baoding 071000,China)

Abstract: MD5 and SHA-1 are the two known Hash Algorithm which are widely applied in information security, they are both developed from MD4.This paper introduces their algorithm logic in detail, compares them though simulation experiment ,software testing, etc. Finally, it draw a conclusion: The SHA-1 algorithm is more secure than the MD5 algorithm, but in the same hardware, SHA-1 is slower than MD5.

Key words: Hash Algorithm;SHA-1;MD5; message authentication; message digest

1 引言

散列算法也叫散列函数、Hash 函数、哈希函数、杂凑函数,在现代密码学中扮演者重要角色^[1]。散列函数是一种公开函数,用于将任意长得消息映射为较短的、固定长度的一个值作为认证符,经常称函数值 H(M)为散列值、哈希值或消息摘要等。从密码角度看,散列函数也可以看做是一种单向密码体制,只有加密过程,不能解密。在密码学和数据安全技术中,散列函数是实现有效、安全可靠数字签名和认证的重要工具,是安全认证协议中的重要模块^[2]。目前在信息安全中,MD5、SHA-1 是当前国际通行的两大散列函数。

2 MD5 算法

MD5 报文摘要算法(RFC 1321)是由 Rivest(公开密钥密码 RSA 算法的设计者之一)所设计的单项散列函数。其中,MD 表示消息的摘要。MD5 不基于任何假设和密码体制,它采用了直接构造的办法,速度很快、非常实用。因此 MD5 曾是使用最为广泛的安全散列算法。MD5 算法实现共需要五个步骤。

第一步,附加填充位。首先填充消息,使其长度为一个比 512 的倍数小 64 位的数。填充方法:在消息后面填充一位 1,然后填充所需数量的 0。填充位的位数为 1~512。

第二步,附加长度。将原消息长度的 64 位表示附加在填充后的消息后面。当原消息长度大于 2^{64} 时,用消息长度 mod 2^{64} 填充(即仅取最低 64bit)。消息长度恰好是 512 的整数倍。

可以表示为 L 个 512bit 的数据块。

第三步,初始化 MD 缓冲区。一个 128bit MD 缓冲区用以保存中间和最终 Hash 函数的结果。它可以表示为 4 个 32bit 的寄存器(A、B、C、D)。寄存器初始化为以下的十六进制值:A=67452301;B=EFCDAB89;C=98BADCFE;D=10325476。

第四步,按 512 位的分组处理输入消息。处理每个消息块,每个消息块可分为 16 个字,记为 M[0],M[1],...,M[15]。这一步为 MD5 的主循环,包括四轮。每个循环都以当前的正在处理的 512bit 分组和 128bit 缓冲值 ABCD 为输入,然后更新缓冲内容。四轮的操作类似,每一轮进行 16 次操作,但每轮访问的数据的次序有所变动。每次操作对 A、B、C 和 D 中的其中三个做一次非线性函数运算,然后将所得结果加上第四个变量,再将所得结果向右位移一个不定的数,并加上 A、B、C 或 D 中之一。最后用该结果取代 A、B、C 或 D 中之一^[3]。每次使用的不同的基本逻辑函数,记为 F、G、H、I。其中,

$$F(B,C,D)=(B\wedge C)\vee(\bar{B}\wedge D)$$

$$G(B,C,D)=(B\wedge D)\vee(C\wedge \bar{D})$$

$$H(B,C,D)=B\oplus C\oplus D$$

$$I(B,C,D)=C\oplus (B\vee \bar{D})$$

第五步,输出结果。所有 L 个 512bit 数据块处理完毕后,由 A、B、C、D 四个寄存器的输出按低位字节在前的顺序得到 128 位的消息摘要。

收稿日期:2016-03-29

作者简介:王泽,男,吉林长岭人,助理工程师,硕士,主要研究方向为计算机网络;曹莉莎,女,助教,硕士。

3 SHA-1 算法

安全哈希算法(Secure Hash Algorithm)主要适用于数字签名标准(Digital Signature Standard DSS)里面定义的数字签名算法(Digital Signature Algorithm DSA)。事实上 SHA-1 目前是全球使用最为广泛的哈希算法,已经成为业界的事实标准。其实现同样共需要五个步骤。

第一步,填充消息。末尾添加一些额外的比特来填充消息,与 MD5 填充方式完全相同,对消息进行填充,使得其消息长度的值模 512 等于 448(448=512-64),若原始消息长度正好是模 512 为 448,则也要增加 1 个 512 比特填充块,也就是说最少要填充 1 个 512 块。填充内容为第一个比特位是 1,其余全部为 0。

第二步,补足长度。在填充消息的末尾添加 64 比特的块,该 64 比特是原始消息二进制表示的长度。如果消息长度变换为二进制块的位的个数小于 64,则左边补 0,使得块的长度刚好等于 64 位。如果消息长度变换为二进制块的位的个数大于 64,则仅取最低的 64 比特,即 mod 2^{64} 。最终,消息长度成为 512 的整数倍。

第三步,初始化缓冲区。初始化 SHA-1 的初始输出,放在 5 个 32 位寄存器 A、B、C、D、E 中,这些寄存器随后将用于保持散列函数的中间结果和最终结果,SHA-1 的 5 个寄存器初始值为 160 比特。IV 初始变量为(十六进制表示):

A = 67452301 , B = EFCDAB89, C =98BADCFE, D = 10325476, E =C3D2E1F0

前 4 个值和 MD5 相同。

第四步,数据处理。消息划分成 512 比特的块,每个块由 16 个 32 位字组成,通过混合和移位,块中的 16 个字被扩充为 80 个字,存放在 $W[k], k=0,1,\cdots,79$ 中。每轮的结构类似,但逻辑函数不同,设分别为 f_1, f_2, f_3, f_4 ,每轮的输入为 512 比特,输出为 160 比特。 f_{1-4} 逻辑函数的定义如下所示:

$$f_1=f(t,B,C,D)=(B\wedge C)\vee(\bar{B}\wedge D) \quad 0\leq t\leq 19$$
$$f_2=f(t,B,C,D)=B\oplus C\oplus D \quad 20\leq t\leq 39$$
$$f_3=f(t,B,C,D)=(B\wedge C)\vee(B\wedge D)\vee(C\wedge D) \quad 40\leq t\leq 59$$
$$f_4=f(t,B,C,D)=B\oplus C\oplus D \quad 60\leq t\leq 79$$

与 MD5 使用的 64 个常量不同,SHA-1 算法在各个回合只有四个常量值,使用数组 K_{1-4} 定义如表 1 所示。

表 1 SHA-1 算法中 K_i 的取值

轮次	回合	输入常数	取值方式
1	$0\leq t\leq 19$	$K_1=5A827999$	$[2^{30}\times\sqrt{2}]$
2	$20\leq t\leq 39$	$K_2=6ED9EBA1$	$[2^{30}\times\sqrt{3}]$
3	$40\leq t\leq 59$	$K_3=8F1BBCDC$	$[2^{30}\times\sqrt{5}]$
4	$60\leq t\leq 79$	$K_4=CA62C1D6$	$[2^{30}\times\sqrt{10}]$

第五步,输出结果。所有 L 个 512 比特数据块处理完毕后,最后第 L 步输出的结果就是 160 比特消息摘要 H(m)。

4 MD5 算法与 SHA-1 算法的比较

由于 MD5 与 SHA-1 均是从 MD4 发展而来,它们的结构和

强度等特性有很多的相似性,但仍有不同之处。

4.1 安全性比较

(1)碰撞率的比较

如果两个输入串的散列值一样,则称这两个串是一个碰撞(collison)。既然是把任意长度的字符串变成固定长度的字符串,所以必有一个输出串对应多个输入串,碰撞是必然存在的^[4]。碰撞率(CR)是已发现碰撞的数量与总共生成的消息的数量之比。通过模拟实验可知,SHA-1 的 CR 值比 MD5 低,这意味着 SHA-1 比 MD5 拥有更高的安全性^[5]。

(2)基于散列值长度的比较

对于同一文件的散列值,长度越长必然会越难破解。MD5 算法散列值为 128 比特,而 SHA-1 算法散列值为 160 比特,MD5 与 SHA-1 的最大区别在于其摘要比 SHA-1 短 32 比特。

(3)已有破解方法比较

MD5 已有密码分析的攻击方法,而 SHA-1 只是理论上被破解,故 MD5 较脆弱。

4.2 执行速度比较

SHA-1 执行需要 80 步,而 MD5 的执行只需要 64 步。而且与 MD5 的 128 比特缓冲区相比,SHA-1 必须处理 160 比特的缓冲区。因此,SHA-1 在相同的硬件上比 MD5 运行的慢。

4.3 综合比较

相同点:MD5 算法与 SHA-1 算法均由 MD4 算法发展而来,属于散列函数。它们均需要五步来完成算法,即填充消息、补足长度、初始化变量、数据处理和最后输出结果。其中前两步完全相同,第四步都需要四轮操作,并且均需要将消息划分为多个 16 字(32bit)即 512 比特消息块来处理。因此它们的结构和算法有很多的相似之处。

不同点:为了方便对比,将两大散列算法列表进行综合比较,如下表 2 所示。

表 2 MD5 与 SHA-1 算法综合比较

	MD5	SHA-1
Hash 值长度	128bit	160bit
步数	64(16×4)	80(20×4)
最大消息长	不限	$\leq 2^{64}$ bit
非线性函数(基本逻辑函数)	4	3(第 2、4 轮相同)
常数个数	64	4
结构形式	Little-endian 格式	Big-endian 格式
安全性(相对而言)	较低	较高
执行速率(在同一硬件上,相对而言)	较快	较慢

5 结束语

MD5 和 SHA-1 是单项散列函数的典型代表,它们广泛地应用在信息安全和数字签名等各个领域。本文主要介绍了两大散列算法——MD5 算法与 SHA-1 算法,通过对这两大散列算法的研究,总结出它们的相同点与不同点。在同一硬件上,SHA-1 比 MD5 运行的要慢。在某种程度上,我们可以得出这样一个结论,当处理的数据很小时,我们可以选择使用 MD5 消息摘要而不是 SHA-1 消息摘要。因为此时 MD5 的碰撞率和运行时间都不高,这可以大大节约资源。(下转第 249 页)

方式,将会有很大的发展前景。

4 结束语

微虚拟化技术的兴起,是大势所趋,随着网络、系统和应用的安全问题不断地出现,给各行业造成的威胁越来越大,保证应用的安全是微虚拟化的主要任务,随着微虚拟化技术的不断完善,我们的信息安全将进一步得到保证,微虚拟化技术将进一步得到应用。

参考文献:

[1] 刘常宏. 浅议虚拟化技术在中小企业信息化中的应用[J]. 中

小企业管理与科技(上旬刊), 2015, (6): 229-230.
[2] 闫龙川,刘志永. 桌面虚拟化技术研究与应用[J]. 电力信息化,2010,07:55-58.
[3]郭涛. 微虚拟化技术将虚拟机效率再提升 30%[N]. 中国计算机报,2012-04-30030.
[4] 刘震宇,吴俊军. 一种基于微内核虚拟化的设备驱动优化模型[J]. 计算机工程与科学,2011,12:44-51.
[5] 陈铨. 基于微内核虚拟化技术的高可靠性嵌入式软件平台研究[D]. 浙江大学,2009.

(上接第 247 页)

参考文献:

[1] 张仕斌,万武南,张金全,等. 应用密码学[M]. 西安:西安电子科技大学出版社,2009.
[2] 胡建伟,马建峰. 网络安全与保密[M]. 西安:西安电子科技大学出版社 2003.

[3] 魏晓玲. MD5 加密算法的研究及应用[A]. 延安:延安大学计算中心,2010.
[4] 刘红军. MD5 防碰撞和穷举变换算法的研究与实现[J]. 包头:职大学报,2008(2).
[5] Ming Hu, Yan Wang. The Collision Rate Tests of Two Known Message Digest Algorithms[C]. 2009 International Conference on Computational Intelligence and Security, 2009.