WAP 中 WTLS 层数据完整性的实现*

汪血焰,陈前斌
(重庆邮电学院,重庆 400065)

摘 要:在 WAP 的 WTLS 层中,通过 HASH 函数实现的消息验证码可以保证数据传输时的完整性,这样可以有效地防止数据在传输过程中被主动攻击者修改,也可以防止数据被重放攻击和拒绝服务攻击。叙述了 HMAC 中 SHA-1 及 MD5 算法的实现,重点分析了这 2 种算法各自的优越性。

关键词:WAP;WTLS;HASH 函数;HMAC;SHA-1;MD5
中图分类号:TN919.3 **文献标识码**:A **文章编号**:1004-5694(2002)02-0057-05

HASH Function Implementation of the Integrity of Data in WAP's WTLS

WANG Xue-yan, CHEN Qian-bin
(*Mobile Communication Engineering Research Center*, *CUPT*, *Chongqing* 400065, *China*)

Abstract: The integrity of data in WAP's WTLS layer is implemented by MAC with HASH function. In this way the transmitting data to be inserted, modified and registered can be efficiently protected, and attacks or denial-of-service attack to be launched by active attacker can be avoided. This paper describes the implementation of SHA-1 and MD5 in HMAC, and analyzes the superiority of the two algorithms.

Key words: WAP; WTLS; HASH function; HMAC; SHA-1; MD5

0 引 言

随着无线技术的发展,客户运用 WAP 上网的趋势将大大增加,而电子商务运用 WAP 进行交易也更加方便。这些服务的前提要求是一个安全保密的数据传输,无线传输层安全标准(WTLS)正提供了这种服务。WTLS 的主要目的就是在客户与 WAP 网关之间提供保密性,数据完整性,鉴权等功能以保证 WAP 安全信息。

在 WAP 协议 WTP 和 WDP 两层间创建了一个 WTLS 层安全通道。它以 TLS(Transport Layer Security,类似 SSL)为基础,为窄带无线信道做了优化,它的主要特点是:

① 使用消息验证编码(MAC:Message Authentication Code)保证数据的完整性;

② 用加密算法保证数据的保密性;

③ 客户与服务器之间使用数字证书(如 WTLS,X.509,X9.68)进行身份验证。

其中,数据的完整性保证接收与发送的数据一致,接受方确认数据没有经过主动攻击者修改,以防止数据破坏甚至拒绝服务攻击。WTLS 完整性保护在数据传输中,一旦发生以上的攻击,接收方将通过 MAC 完整性验证得知数据被破坏,并向发送方发出告警。根据告警程度(致命、严重、一般),采取终止双方数据传输或重传加密数据,防止以上攻击产生的损失。

* 收稿日期:2001-12-11
作者简介:汪血焰(1971-),男,四川省乐山市人,重庆邮电学院 99 级硕士研究生,目前主要从事网络安全研究。

1　HASH 函数与 HMAC 的基本概念

1.1　HASH 函数

哈希函数，即对于任意长度的信息 x 进行哈希函数运算后，压缩成固定长度的数 M。哈希函数必须具有以下特性：

① 能够应用到任何大小的数据上；

② 能够生成长度固定的输出；

③ 处理任意给定的长度信息 x，$H(x)$的计算相对简单，使得硬件和软件的实现可行；

④ 对于任意给定的代码 h，要发现满足 $H(x)=h$ 的 x 在计算上是困难的；

⑤ 对于任意给定的块 x，要发现满足 $H(y)=H(x)$而 $y\neq x$ 的 y 是困难的；

⑥ 要发现满足 $H(x)=H(y)$对在计算上是困难的。

前 3 个特点是哈希函数在数据验证的实际应用中所需要的。第④个特点是“单向”特性：数据很容易正向生成验证码，但验证码反向很难恢复原数据。第⑤个特点保证：对给定数据是很难用替换数据生成相同的哈希值。仅满足①～⑤特性的哈希函数为弱哈希函数。第⑥个特性可以防止诸如“生日攻击”等复杂类型的攻击，满足第⑥特性的哈希函数为强哈希函数。

1.2　HMAC

从哈希函数(如 SHA-1 与 MD5)中开发 MAC 越来越热，HMAC 已作为 RFC2104 发布，而且选作 IP 安全代理强制实施的 MAC，并应用到 WAP 协议的 WTLS。

1.2.1　HMAC 的设计目标

RFC2104 中列出的 HMAC 的设计目标为：

① 可不作修改就能使用现有的哈希函数；

② 在需要更快或更安全的哈希函数时，允许对嵌入的哈希函数作简单的替换；

③ 能保持哈希函数的初始性能，而不会大幅降低性能；

④ 可用简单的方式使用和处理密钥；

⑤ 在对嵌入哈希函数合理假设的基础上，可以对验证机制强度进行易于理解的加密分析。

最后一项设计目标是 HMAC 优越于以哈希为基础的其他方案的最主要的优点。已经证实，HMAC 具有足够的安全性，嵌入哈希函数拥有一定抗攻击强度。

1.2.2　HMAC 的计算

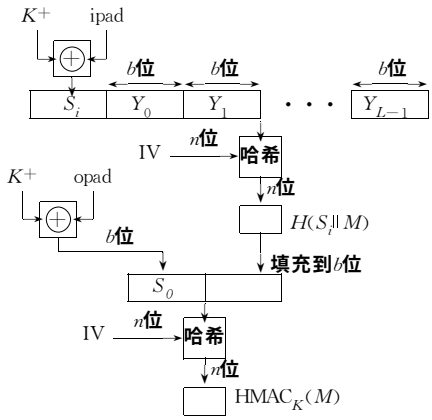

图1　HMAC的结构

Fig. 1 HMAC structure

图 1 中描述了 HMAC 的整个操作，其中各符号定义为：H 为嵌入的哈希函数(如 SHA-1 或 MD5)；M 为输入到 HMAC 的消息(在嵌入的哈希函数中指定的填充内容)；Y_i 为 M 的第 I 个块，$0\leqslant i\leqslant(L-1)$；$L$ 为 M 中块的总数；b 为块中的位数；n 为嵌入哈希函数生成的哈希码的长度；K 为保密密钥，如果长度比 b 大，密钥就输入到哈希函数中，生成 n 位的密钥，推荐长度为$\geqslant n$；K^+为 K 的左面补零，结果的长度为 b 位；ipad 表示 00110110(16 进制的 36)，重复 $b/8$ 次；opad 表示 01010011(16 进制的 5C)，重复 $b/8$ 次。

HMAC 可以表达为：

$$HMAC_K = H[(K^+ \oplus opad) \| H[(K^+ \oplus ipad) \| M]]$$

其含义为：① 左端添零，创建 b 位的 K^+串(例如，如果 K 的长度为 160 位，$b=512$ 位，将会给 K 增加 44 个零字节 0x00)；② K^+与 ipad 进行异或(逐位异或)，生成 b 位的块 S_i；③ 将 M 附加到 S_i 上；④ 将 H 应用到第③步的生成流中；⑤ 将 K^+与 opad 进行异或，生成 b 位的块 S_0；⑥ 将第④步的哈希结果附加在 S_0 上；⑦ 将 H 应用到第⑥步的生成流上并输出结果。

ipad 异或会导致 K 有一半的位翻转。与此类似，opad 异或也会导致 K 有一半的位翻转，但与前者是不同的一组位。实际上，通过哈希函数处理 S_i 和 S_0 时，已经用 K 生成了 2 个密钥。

2 SHA-1 与 MD5 哈希函数算法

2.1 SHA-1 算法

安全哈希算法(SHA)是由美国国家标准技术局(NIST:National Institute of Standards Technology)开发出来,并在 1993 年作为联邦消息处理标准(FIPS PUB180)公布。在 1995 年改进成版本 FIPS PUB 180-1,通常叫做 SHA-1。SHA-1 相对 SHA 增加了左移循环操作。

SHA-1 算法以最大长度不超过 2^{64} 位的消息为输入,生成 160 位的消息摘要输出。用 512 位块处理输入。图 2 画出生成摘要的整个消息处理过程。处理包括以下 5 步。

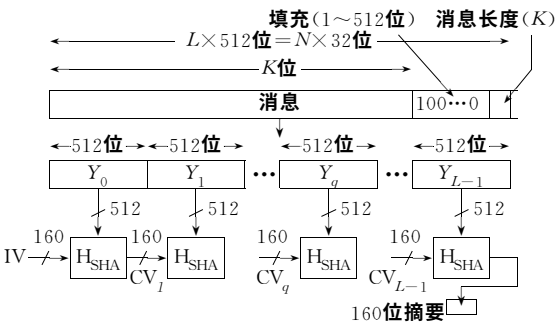

图2 用SHA-1生成消息摘要图

Fig. 2 Message abstract generated by SHA-1

第 1 步:附加填充位。使填充消息的长度等于 448 mod 512,即填充消息的长度为 64 位。填充必须要加的,即使消息的长度完全达到了想要的长度。因此,填充位的数目在 1 到 512 的范围之内。填充的构成先是一个 1,后跟必要数目的 0 位。

第 2 步:附加长度。将 64 位块附加到消息上,块被当作无符号 64 位整数来处理(高位字节优先)而且包括原始消息的长度(填充前)。附加的长度数值将使得填充攻击更加困难,而扩展消息的总长度为 $L\times512$ 位,同样也是 16 个 32 位字的倍数,可以用 $M[0\cdots N-1]$ 表示结果消息的字节,N 是 16 的整数倍。因此,$N=L\times16$。

第 3 步:初始化 MD 缓冲区。可以用 160 位来存放哈希函数的中间和最终结果。缓冲区可以表示成 5 个 32 位的寄存器(A,B,C,D,E)。这些寄存器都已被初始化为以下 32 位的整数(16 进制数):

H0=67452301;H1=EFCDAB89;H2=98BADCFE;H3=10325476;H4=C3D2E1F0

第 4 步:处理 512 位(16 字节)块的消息。此算法的核心是压缩函数模型,它包括了 4 个处理过程,每个过程都有 20 步。图 3 中列出了逻辑图。4 个过程具有类似的结构,但每个使用的初始逻辑非线性函数都不同,分别为 f_1、f_2、f_3、f_4,其操作:

f_1=(B AND C) OR ((NOT B) AND D) $(0\leqslant t\leqslant 19)$

f_2=B XOR C XOR D $(20\leqslant t\leqslant 39)$

f_3=(B AND C) OR (B AND D) OR (C AND D) $(40\leqslant t\leqslant 59)$

f_4=B XOR C XOR D $(60\leqslant t\leqslant 79)$

注:AND 表示逻辑与;XOR 表示逻辑与异或;NOT 表示逻辑非;OR 表示逻辑或。

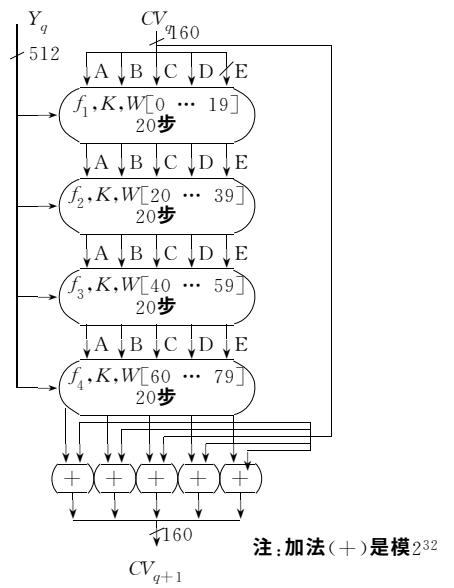

图3 单个512位块的SHA-1处理逻辑图

Fig. 3 SHA-1 logic processing of single 512 chip

每个过程都以当前处理的 512 位块(Y_q)和 160 位的缓冲区数值 ABCDE 为输入(见图 4),并对缓冲区的内容进行更新。每个过程也利用附加常量 K_t,其中 $0\leqslant t\leqslant 79$,表示 5 个过程中的 80 步之一。事实上只使用了 4 个不同的常量,见表 1。

将 512 位块分为 32 位的 16 个字:$W(0)$,$W(1)$,…,$W(15)$,其中 $W[0]$为最左边的字。

(1) For t=16 to 79 do

W(t)=S_1(W(t-3)) XOR W(t-8) XOR W(t-14) XOR W(t-16))

A=67452301 B=EFCDAB89

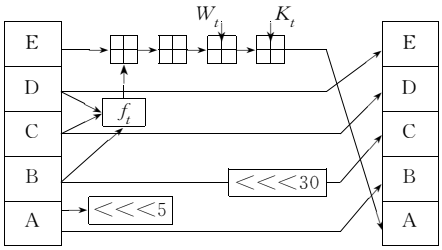

图4　SHA−1第t次运算

Fig. 4 SHA−1 operation at t−times

C=98BADCFE　D=10325476

E=C3D2E1F0

(2) 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;

A=H0+A,H1=H1+B,H2=H2+C,

H3=H3+D,H4=H4+E。

注:$S_n(X)=(X<<n)$ OR $(X>>32-n)$,**表示** 32 **位的** X **字左移循环** n **位,是** SHA-1 **对** SHA **的改进。**$S_n(X)$ **表示为**$<<<n$.

将第 4 **个过程的输出添加到第一个过程的输入中,生成** CV_{q+1}**。加法过程独立进行,将** CV_q **中的相应字加到** 5 **个缓冲区的** 5 **个字上,加法模数为** 2^{32}**。**

第 5 **步:输出** 160 **位散列。**

表 1　**数值的** 16 **进制和** 10 **进制表示值**

Tab. 1 Display of values in decimal system and hexadecimal system

步骤号	16 进制	获取整数部分
$0\leqslant t\leqslant 19$	K_t=5A827999	$[2^{30}\times\sqrt{2}]$
$20\leqslant t\leqslant 39$	K_t=6ED9EBA1	$[2^{30}\times\sqrt{3}]$
$40\leqslant t\leqslant 59$	K_t=8F1BBCDC	$[2^{30}\times\sqrt{5}]$
$60\leqslant t\leqslant 79$	K_t=CA62C1D6	$[2^{30}\times\sqrt{10}]$

2.2　MD5 算法

MD5 **是** Ron Rivest **对** MD4 **的改进,改** MD4 **的** 3 **轮操作为** 4 **轮(见图** 5**)。以** 512 **位分组来处理文本,每一分组又分为** 16 **个** 32 **位分组。输出由** 4 **个** 32 **位分组组成** 128 **位散列值。**MD5 **的计算分为** 5 **步进行。**

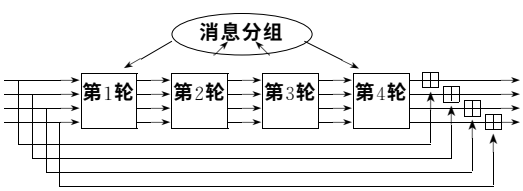

图5　MD5主循环

Fig. 5 Major circulation of MD5

第 1 **步:与** SHA-1 **的第** 1 **步相同。**

第 2 **步:与** SHA-1 **的第** 2 **步相同。**

第 3 **步:初始化** MD **缓冲区。可以用** 160 **位来存放哈希函数的中间和最终结果。缓冲区可以表示成** 4 **个** 32 **位的寄存器**(A,B,C,D)**。这些寄存器都初始化为** 32 **位的整数**(16 **进制数**): A=0x01234567; B=0x89abcdef; C=0xfedcba98; D=0x76543210.

第 4 **步:**4 **个非线性函数,输入** 3 **个** 32 **位的字,输出一个** 32 **位的字:**

F(A,B,C)=(A AND B)OR ((NOT A)AND C)

G(A,B,C)=(A AND C)OR (B AND(NOT C))

H(A,B,C)=A XOR B XOR C

I (A, B, C) =B XOR(A OR(NOT C))

将 512 **位块分为** 32 **位的** 16 **个字:**M(0),M(1),…,M(15)**;操作如下:**

AA = A;BB = B;CC = C;DD = D;

/* [abcd k s i]**的参数排列顺序参考** RFC1321;/

第一轮:

a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s).

第二轮:

a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s).

第三轮:

a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s).

第四轮:

a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s).

/* **初始化常数加上操作后的值** */

A = A + AA; B = B + BB;C = C + CC;D = D + DD

注:其中 $T[i]$**为** $2^{32}\times\mathrm{abs}(\sin(i))$**的整数部分,**$i$ **的单位为弧度;**$<<<s$ **表示左移循环;**$X[k]=M[16q+k]$**在第** q **个** 512 bit **分组的第** k **个** 32 bit **字。**

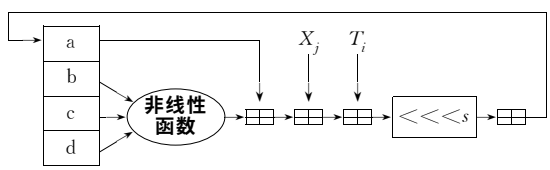

图6　MD5的一个执行过程

Fig. 6 Impementation process of MD5

第 5 **步:输出** 128 **位散列,以** 32 **位字** A,C,D,B **先后顺序。**

2.3 SHA-1 与 MD5 的安全性对比

(1) MD5 在 MD4 基础上增加了第 4 轮,每轮使用相同的非线性函数,重复使用相同的加法常数。

(2) SHA-1 延续了 SHA 中第 4 轮($t=60\sim79$)与第二轮($t=20\sim39$),每轮使用相同的非线性函数,重复使用相同的加法常数。

(3) MD5 为减弱第 2 轮中函数 G 的对称性,将 MD4(B AND C) OR (B AND D) OR (C AND D) 变为(A AND C)OR (B AND(NOT C)),而 SHA-1 采用了 MD4 的形式。

(4) MD5 与 SHA-1 每一步加上了上一步的结果,这将引起更好的雪崩效应。MD5 采用 4 个变量,SHA-1 增加了第 5 个变量 E,而不是非线性函数中已经引用的 B,C,D,这使 den Boer-Bosselaers 攻击对 SHA-1 不起作用。

(5) MD5 改变了第 2 轮和第 3 轮中消息分组的次序,使其形式更不相似,而 SHA-1 完全不同,采用了循环纠错编码。

(6) MD5 与 SHA-1 采用近似优化了每一轮中的循环左移,以实现更快的雪崩效应,各轮的位移量互不相同。

3 结 论

SHA-1 与 MD5 是在 MD4 基础上改进发展的,SHA-1=MD4+扩展转换+附加轮+更好的雪崩效应;MD5=MD4+改进的位运算+附加轮+更好的雪崩效应。SHA-1 由于产生 160 位散列,而 MD5 为 128 位散列,此时生日攻击仅有约 2^{64} 次杂凑;同时 MD5 对差分分析的抵抗能力弱。因此 SHA-1 相对 MD5 具有更好的抗攻击能力,特别是更能抵抗穷举攻击(包括生日攻击)。在 WTLS 中,通过 HMAC 来实现数据传输的完整性,并因 SHA-1 比 MD5 有优越的抗攻击能力及良好的雪崩效应,所以,WTLS 中优先考虑了 SHA-1。

WTLS 考虑到移动通信的特点,推荐可以取哈希函数 SHA-1 或 MD5 的前 10 或 5 字节为输出(其中 10 字节输出优先于 SHA-1),但这样也减少了安全强度。完整性验证中涉及到使用 Diffie-Hellman 密钥交换体制协商的密钥生成消息验证码 MAC,MAC 并附在消息数据块后面。在这种技术中,通信双方(A 与 B)共享密钥 K_{AB}。如果 A 有数据要送给 B,它就会计算出 MAC,它是数据和密钥为输入的布尔函数:$MAC=F(K_{AB},M)$。数据和 MAC 被传送给接收方。接收方对接收到的数据进行同样的计算,用协商好的 K_{AB} 密钥生成新的验证码 MAC,并把接收的代码与计算出的代码相比较。如果只有接收方和发送方知道密钥,而且接收的代码与计算出的代码才相吻合。WTLS 中具有如下 3 个功能。

① 接收方能够根据 MAC 确信消息没有经过改动。如果攻击者改动了消息,但没有改动 MAC,则接收方计算出的与接收到的 MAC 不同。因为攻击者不知道密钥及 HASH 函数的单向性,所以攻击者也就不能在数据中改动相应的 MAC.

② WTLS 协议中的生成 HMAC 的密钥运用了数据包的序列号,则接收方能确定序列号,攻击者很难成功地改动序列号,并同时改动数据。

③ WTLS 协议中双方使用 HMAC 的密码动态更新,加强了 MAC 的强度。因此 WTLS 在 HMAC 中运用 HASH 函数保证了各级的数据完整性,防止多种攻击。

参 考 文 献

[1] WTLS-20010406-p. Wireless Application Prorocol- WAP-261[S]. 2001.

[2] RFC2104. HMAC: Keyed-hashing for Message Authentication[S]. 1997.

[3] RFC1321. The Message Digest Algorithm [S]. 1992.

[4] FIPS PUB 180-1. SECURE HASH STANDARD[S]. 1997.

[5] William Stallings. 网络安全要素—应用与标准[M]. 北京:人民邮电出版社,2000.

[6] Bruce Schneier. 应用密码学[M]. 吴世忠译. 北京:机械工业出版社,1999.

(编辑:龙能芬)