构造基于 Logistic 映射的 Hash 函数

游中胜[1] 刘 锋[2]

(重庆师范大学数学与计算机学院 重庆 400047)[1] (重庆大学计算机学院 重庆 400044)[2]

摘 要 单向 Hash 函数是实现有效、安全可靠数字签名和认证的重要工具,是安全认证协议的重要模块。本文针对文[5]中基于混沌映射的 Hash 算法的一些缺陷,提出了一种解决方案和算法,并对该算法进行了仿真实验,还进行了混乱与散布性质统计分析,验证了本文所提出的算法的有效性。

关键词 混沌映射,Hash 函数,混乱与散布

Constructing a Hash Function Based on Logistic Mapping

YOU Zhong-Sheng[1] LIU Feng[2]

(College of Mathematics and Computer Science, Chongqing Normal University, Chongqing 400047)[1]

(College of Computer, Chongqing University, Chongqing 400044)[2]

Abstract The one-way function of hash is an important tool of realizing effective and safe and credible numerical signature and authentication, it is an important part of safe authentication protocol. According to some shortcomings of a chaos-based hash algorithm presented in document[5], this paper puts forward a kind of solution project and algorithm, and does the emulational experiment to the algorithm, and also conducts a statistical analysis of the improved algorithm in confusion and diffusion properties, and therefore verifies the efficiency of the algorithm.

Keywords Chaotic mapping, Hash function, Confusion and diffusion

1 引言

随着电子商务的快速发展,如何保证交易的安全越来越为人们所关注。以公钥密码算法、数字签名等为代表的加密安全技术也随之成为研究的热点。单向 Hash 函数是实现有效、安全可靠数字签名和认证的重要工具,是安全认证协议的重要模块,常见的单向 Hash 函数有:Snefru 算法[1]、N-Hash[2]、MD4[3]、MD5[4]等。研究表明,Snefru 算法、N-Hash 在抗击差分密码分析方面存在缺陷。目前运用较多的是 MD5,其中 MD5 是在 MD4 基础上改进而来的。近年来,混沌理论得到了密码学界的广泛关注,开始将混沌映射运用到 Hash 函数的构造上。文[5]中提出了一种利用 Logistic 映射产生单向 Hash 函数的算法,该算法存在一些缺点:(1)由于算法在经过前几轮的迭代后才进入混沌状态,这样使得算法在对 ASCII 码值相近的字符进行 Hash 时,前面几位是相同的。例如对'A'和'B'计算 Hash 值时就会出现这种现象;(2)从算法中可以看到各个明文块 Hash 时明显缺乏关联,这样会造成在对字符串'AB'加密时,散列值的前几位全为 0。原因在于'A'与'B'的 ASCII 码值相近,在对它们单独求 Hash 值时,前几位是相同的,由于算法中最后只是简单地将每个明文块的 Hash 值模 2 相加,因此会出现前几位全 0 的情况,这样大大降低了算法的安全性;(3)当明文中最后一位发生改变时,得到的 Hash 值结果将不会发生大的改变,这是因为最后一位改变时,对前面的明文块没有产生影响,只是在最后模 2 相加时,对结果产生了局部影响。这一点上没有很好地满足 Hash 函数必须具备的第 1 个特征。本文针对该算法的一些安全缺陷,提出了对该算法的改进方案,还对此方案安全性做了相应的分析。

2 一种改进的基于 Logistic 映射的 Hash 算法

2.1 改进思路

针对上述存在的问题,对算法做了以下改进:1. 针对算法需要在经过前几轮迭代才能进入混沌状态的弱点,现采用随机生成 X 的初值。X 的初值=Random(-1,1) 这样即使混沌状态进入比较晚,仍然能够保证在对 ASCII 码值相近的字符求散列值时,结果出现很大的不同;2. 针对各个明文块 Hash 时缺乏关联,现增加了上下文的关联,关联的方法是将前一个明文块求得的 X 序列中的最后一个 X 的值作为下一个明文块的输入。不妨设第 n 个明文块的 X 的初值设为 $X_{(n)init}$,第 n 个明文块 Hash 后得到散列值序列的最后一位设为 $X_{(n)32}$,那么关联规则可以用下式表示:

$$X_{(n)init}=X_{(n-1)32}$$

其中 $n>1$;3. 针对算法中存在的第(3)个问题,提出了一种解决办法:设定一个 p 值,这个值决定了 Hash 的次数,当 $p>1$ 时,就将对第一次求得的散列结果再进行($p-1$)次 Hash 运算,这样增强了 Hash 结果的混乱性,达到了 Hash 的效果;4. 算法 Hash 结果从原来的 128 位扩展到 160 位,进一步提高了算法精度,增强了函数的抗攻击能力。

2.2 算法描述

步骤 1 将明文消息分块,在本实现方法中是以一个字符作为一个明文信息块。

步骤 2 将每个明文信息块的每一块量化,在本实现方法中采用求每个字符的 ASCII 码值,这样量化的值域是[0,255]。

游中胜 讲师,硕士生,主要研究方向:计算机网络、算法、数据库应用。

步骤3 将量化后的值映射到某个值域，使得用该集合内的值作为参数能够使混沌系统处于混沌状态，本实现中采用的映射：

$\mu_i = 1.746 + 0.001 * \mathrm{Asc}(M_i)$，指定 Hash 次数 p。

步骤4 随机分配[-1,1]之间的随机数，作为 X 的初值，将 μ_i 代入 Logistic 映射中经过 32 次的迭代可得到序列 $\{X_i\}$。

步骤5 将 X_i 通过一定方式转换成 5 位二进制序列 Y_i，Y_i 对应 5 位二进制位。保存 Y_i 的值。

步骤6 将序列 $\{X_i\}$ 中最后一个 X 即 $X32$ 的值作为下一个明文信息块的输入，转到步骤 3 执行，直到每个明文信息块都得到对应的 Y_i。

步骤7 将每 Y_i 所得的二进制序列逐位模 2 相加可得该明文消息的 Hash 值(共 160 位)。

步骤8 若 $p>1$，则对得到的 Hash 值再进行 $(p-1)$ 次 hash 运算，从而得到最终的 Hash 值。

3 仿真与分析

3.1 文本 Hash 结果

根据上述算法编制了程序，并作了相关测试：

取初始文本 1 为“Find solutions specific to your business, your industry - fast.”，文本 2 将文本 1 中首字母 F 改为 G，文本 3 将文本 1 最后的句号改为逗号，文本 4 将文本 1 中的逗号改为句号。调整的原则是保证后面 2 个文本相对于文本 1 只有 1 bit 的变化。同时分别对 $p = 1,2,3$ 作测试。Hash 结果用二进制表示，下面列出当 $P=3$ 时，得到的 Hash 结果。

文本 1：

00000100100011011011111100101000000001010011010000110101100101000011101111111011000010101000111011111010011111000010011111101011100101011100110111010000000010010111

文本 2：

11001000100110110011101100100000011011100110010101111000010010010011000011110001110101011010000101110111100101100010100010011111100011001110000010001011101000000100000011010010000010010010101111101000101010101

文本 3：

00100010110101111011110111111001011100100100000100000100111010101100100110110110101100100100100001000101110100000010000001101001000001001001010111110100001010101

文本 4：

11111110001001001101001111010011011100010111010101011111110011000010011100100101111100101110010010100001101101100110101001001110000010100110101111010110110101111011

可见该算法的单向 Hash 性能很好，初值的每 bit 变动，结果都会发生很大的变化，具有很高的初值敏感性。

3.2 混乱与散布性质统计分析

Hash 函数要尽量做到相关明文对应的散列值不相关，而对于结果的二进制表示，每 bit 只有 0 或者 1 两种可能，因此理想散列值的散布效果应该是初值的微小变化将导致结果的每 bit 都以 50%的概率变化。考察算法在明文发生 1bit 变化的情况下，引起 Hash 密文结果的变化比特数。

定义平均变化比特数：$\bar{B}=\frac{1}{N}\sum_{n=1}^{N_1}B_n$，

定义平均变化概率：$P=(\bar{B}/160)\times 100\%$，

B 的均方差：$\Delta B=\sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(B_n-\bar{B})^2}$，

P 的均方差：$\Delta P=\sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(B_n/160-p)^2}$

其中 N_i 为统计次数，B_n 为第 n 次测试时结果的变化 bit 数，每次测试方法为：在明文空间中随机选取一段明文进行 Hash，然后改变明文块 1bit 的值得到另一 Hash 结果，比较两个结果得到变化比特数 B_n。

表 1 为统计出的 3.1 节 4 个文本中位置相同而值不同的数目。

表 1 Hash 值比较

比较	文本 1	文本 2	文本 3	文本 4
文本 1				
文本 2	76			
文本 3	79	85		
文本 4	83	83	78	

经计算，$\bar{B}=80.7$，$P = 50.42\%$，$\Delta B = 3.5$，$\Delta P = 2.236\%$。从以上结果可以看出，初值的微小变化使散列值的结果有将近一半的位改变，算法的平均变化比特数 $\bar{B}$ 和每比特平均变化概率 P 都已非常接近理想状态下的 80bit 和 50%的变化概率，相当充分和均匀地利用了密文空间，从统计效果来看，攻击者在已知一些明文密文对，对其伪造或反推其它明密文时没有任何帮助，因为明文的任何细微变化，密文从统计上来看在密文空间中都是接近等密度地均匀分布，从而得不到任何密文分布的有用信息。ΔB 和 ΔP 标志着 Hash 混乱与散布性质的稳定性，越接近 0 就越稳定，从实验结果看该算法 ΔB 和 ΔP 已很小，可见该算法可有效地对数据进行保护。同时，本算法产生 160 位散列值，所以它比其它 128 位散列函数更能有效抵抗穷举攻击。

结论 本文针对文[5]中基于混沌映射的 Hash 算法的一些缺陷，提出了一种改进算法，并对该算法进行了仿真。仿真结果表明改进后的算法很好地满足了 Hash 函数所必须具备的各种特性[6]：(1)对于不同的报文不能产生相同的散列码，改变原始报文中的任意一位数值将产生完全不同的散列码；(2)对于任意一个报文无法预知它的散列码；(3)无法根据散列码倒推报文，因为一条报文的散列码可能是由无数的报文所产生；(4)Hash 算法是公开的，不需要保密，它的安全性来自它产生单向散列的能力；(5)散列码有固定的长度，一个短报文的散列与百科全书的散列将产生相同长度的散列码。因此在安全性能上得到了提高。同时，该算法易于实现，是一种快速实用的 Hash 函数构造方案。

参 考 文 献

1 Schneier B. Applied Cryptography, 2nd Edition. Wiley, New York, 1996

2 Miyaguchi S, Ohta K, Iwata M. 128- bit HashFunction (N-Hash). NTT Review, 1990, 2(6)

3 Rivest R L. The MD4 Message Digest Algorithm. RFC1186, Oct. 1990

4 Rivest R L. The MD5 Message Digest Algorithm. RFC 1321, Apr. 1992

5 陈志德，黄元石. 混沌型单向散列函数. 福州大学数学系，2001

6 陈世永. 网络安全原理与应用. 科学出版社，2003

7 Kwok-wo wong. A combined chaotic cryptographic and hashing scheme. Hong Kong, 2002