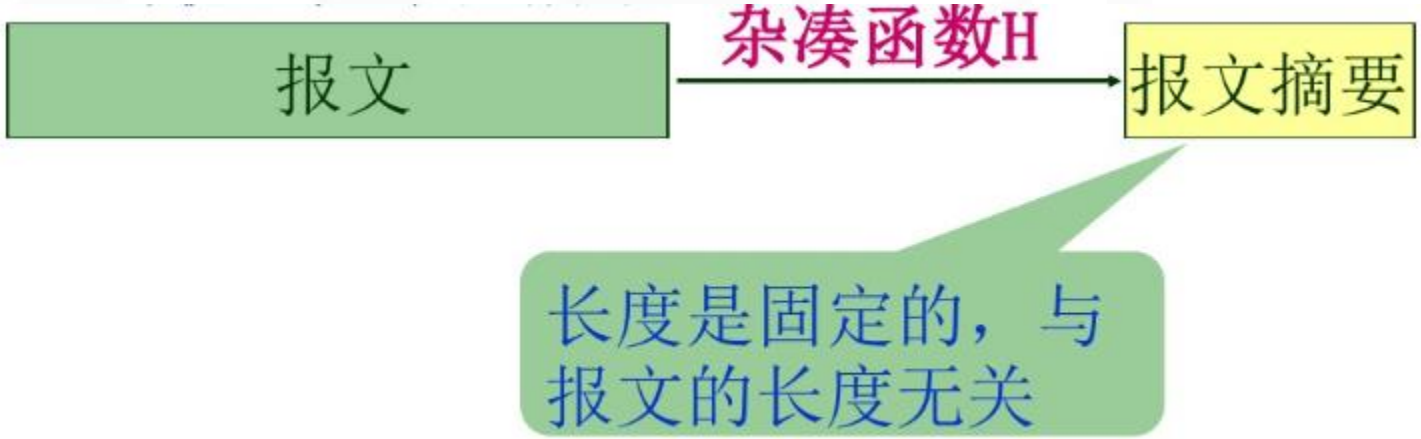
第七章 哈希函数

7.1 哈希函数的概念

7.1.1 哈希函数的概念

- 哈希算法又称杂凑函数、Hash函数、消息摘要函数等。
- 其目的是将任意长度的消息m压缩成指定长度的数据$H(m)$。其中$H(m)$又称为m的摘要或指纹。



7.1.2 哈希函数的应用

- **完整性认证**：$(m, H(m))$。m的任何$H(m)$改变都将导致哈希值$H(m)$的改变，需要完整性认证时，只需计算$H(m)$并与存储的$H(m)$相比较即可。
- **数字签名**：$(m, \text{sig}(H(m)))$**实现真实性**。通常用公钥算法进行数字签名时，一般不是对m直接签名，而是对哈希值$H(m)$签名，这样可以减少计算量，提高效率。

7.1.3 哈希函数应满足的条件

- H能够应用到任何大小的数据块上；
- H能够生成大小固定的输出；
- 对任意给定的m，$H(m)$的计算相对简单，使得硬件和软件的实现可行。
- 单向性（第一原像不可求），对于任意的h，要发现满足$H(m)=h$的m是计算上不可行的；
- 弱抗碰撞性，对于任意给定的m_1，要找到满足$H(m_2)=H(m_1)$，且$m_2 \neq m_1$的m_2，是计算上不可行的；
- 强抗碰撞性，要发现满足$H(m_1)=H(m_2)$，而$m_1 \neq m_2$的对(m_1, m_2)是计算上不可行的。

7.1.4 哈希函数的优缺点

由于消息摘要不包含消息持有者的秘密信息，故：

- 优点：任何人都可对消息的“指纹”进行检验。
- 缺点：掌握消息的人都可生成报文的“指纹”。

上述缺点导致当将消息及其指纹放在一起时，只能检验出消息无意的修改和错误，不能检验出有意的篡改或伪造。

7.2 基本攻击方法

7.2.3 碰撞攻击

- 目的：构造报文m_1和m_2使得$H(m_1) = H(m_2)$
- 生日悖论：23个人的生日互不相同的概率是多少？并不是23/365。
- 所有人生日都不相同的概率为：$\frac{365}{365} \times \frac{365-1}{365} \times \cdots \times \frac{365-22}{365} = 1 \times \left(1 - \frac{1}{365}\right) \times \cdots \times \left(1 - \frac{22}{365}\right) < e^{-1/365} \times \cdots \times e^{-22/365} = e^{-23 \cdot 22/2 \cdot 365} \approx 0.499998$
- 故23个人中至少有两个人生日相同的概率超过$\frac{1}{2}$

7.2.3 碰撞攻击

- 碰撞攻击算法：

（1）随机选取N个报文$m_1, m_2, \ldots, m_N$；

（2）以这N个报文作为杂凑函数的输入，计算出相应的杂凑值，得到集合$S = \{(m_k, H(m_k)) | k = 1,2, \ldots N\}$；

（3）根据$H(m_k)$的大小，对集合S利用快速排序算法重新排序：在排序过程中，如果找到了$H(m_k) = H(m_t)$的两个不同元素m_k和m_k，就将(m_k, m_t)作为结果输出，算法中止；如果找不到，就报告碰撞攻击失败，算法中止。

7.3 MD5哈希算法

7.3 MD5算法

MD填充技术，是将任意长度的消息$M = \{M_1, M_2, \ldots, M_n\}$的最后一个分组$M_n$设置为“真正消息”的长度，这个过程称为MD填充。

MD4是Ron Rivest于1990年设计的，MD5是MD4的改进形式。二者的设计思想相似。

- MD5的特点:

对任意长度的输入，都能产生128位的输出；其安全性不依赖于任何假设，适合高速实现。

7.3 MD5算法

- MD5的安全分析现状：

2004年夏，山东大学的王小云宣布找到使MD5的杂凑值相同的两个消息，这两个消息的差是一个特殊的值，但没有公开构造方法。

之后，王小云教授公布了她的方法。人们后来发现，找出MD5的一个碰撞在PC机是容易的事情。

由于能产生碰撞的消息未必有实际的意义，而且按照王小云教授的方法构造的两个消息都不能人为地控制，因此该攻击并不对MD5造成实际的威胁。

7.3 MD5算法

- 1. 初始化处理：消息填充。

目的是使MD填充后的消息长度是512的整数倍。

方法：设原始消息x的长度是L比特。

（1）求出$d \geq 0$，使得$L+1+d+64$是512的整数倍；

（2）在原始报文x后面添加一个1，然后添加d个0，

最后将消息的长度L用64比特表示，加在最后。

填充后的报文$= x||1||0^d||L$

由$L+1+d+64 \equiv 0 \ mod\ 512$得，$d = -(L+65) \ mod\ 512$

7.3 MD5算法

- MD5填充的例子：

  设x是具有20768比特的长信息，则：

$$
\begin{aligned}
d &= -(L+65) \bmod 512 \\
&= -(20768+65) \bmod 512 \\
&= -20833 \bmod 512 \\
&= -(40\times 512+353) \bmod 512 \\
&= -353 \bmod 512 \\
&= 159
\end{aligned}
$$

故应在x后面添加1个1和159个0，最后再添加原始消息长度20768的64位表示。

7.3 MD5算法

- MD5算法的输入消息x被分成512比特的消息块$x_1, x_2, \ldots, x_t$, t的取值是填充后消息的512比特分组的数目。然后将每个消息块划分成16个32比特的子块，记为$M = M_0 M_1 \ldots M_{15}$, 其中$|M| = 512$, $|M_i| = 32, i = 0,1, \ldots, 15.$

7.3 MD5算法

- 2. 初始向量:

  4个32比特的初始向量：A=0x01234567

  B=0x89abcdef

  C=0xfedcba98

  D=0x76543210

7.3 MD5算法

- 3. 基本运算：

算法主循环用到的四个基本函数为：

$$f(X,Y,Z)=(X\wedge Y)\vee(\overline{X}\wedge Z)$$
$$g(X,Y,Z)=(X\wedge Z)\vee(Y\wedge\overline{Z})$$
$$h(X,Y,Z)=X\oplus Y\oplus Z$$
$$I(X,Y,Z)=Y\oplus(X\vee\overline{Z})$$

X,Y,Z都是32位字。

$x\wedge y$：逐位与运算　　$x\oplus y$：逐位模2加运算

$x\vee y$：逐位或运算　　$\overline{x}$：逐位取补运算

$x+y$：模2^{32}位加运算　　$x\prec\prec s$：循环左移s位

7.3 MD5算法描述

- 轮函数模型：$H_i = h(H_{i-1}, M_i) \oplus H_{i-1}$，其中$H_{i-1} = (A, B, C, D)$，$h$由$round1, round2, round3$和$round4$组成。

具体过程：对$i = 0, \ldots, {}^{N}/_{16} - 1$, 依次执行以下步骤：

（${}^{N}/_{16}$是512比特块的个数）

Step1 令$X[0] = M[16i], X[1] = M[16i] + 1, \ldots, X[15] = M[16i + 15]$

Step2 令$AA = A, BB = B, CC = C, DD = D$;

Step3 执行$(A, B, C, D) = round1(A, B, C, D, X[0], \ldots, X[15])$;

Step4 执行$(A, B, C, D) = round2(A, B, C, D, X[0], \ldots, X[15])$;

Step5 执行$(A, B, C, D) = round3(A, B, C, D, X[0], \ldots, X[15])$;

Step6 执行$(A, B, C, D) = round4(A, B, C, D, X[0], \ldots, X[15])$;

Step7 $A = A + AA, B = B + BB, C = C + CC, D = D + DD$;

最后输出128比特散列值：$MD5(X) = A||B||C||D$

7.3 MD5算法描述

- 函数$round1$的结构：用[a b c d i s t]表示运算

$a \leftarrow (b + [a + f(b,c,d) + M[i] + t_i]) \lll s$（仅替换$a$）

t_i是$2^{32} \times ABS(sin(i))$的整数部分

用$round1$依次执行下述16层运算：

a	(b, c, d)	i	s	t
[A	B C D	0	7	t_1]
[D	A B C	1	12	t_2]
[C	D A B	2	17	t_3]
[B	C D A	3	22	t_4]
[A	B C D	4	7	t_5]
[D	A B C	5	12	t_6]
[C	D A B	6	17	t_7]
[B	C D A	7	22	t_8]

a	(b,c,d)	i	s	t
[A	B C D	8	7	t_9]
[D	A B C	9	12	t_{10}]
[C	D A B	10	17	t_{11}]
[B	C D A	11	22	t_{12}]
[A	B C D	12	7	t_{13}]
[D	A B C	13	12	t_{14}]
[C	D A B	14	17	t_{15}]
[B	C D A	15	22	t_{16}]

7.3 MD5算法描述

- 函数$round2$的结构：用[a b c d i s t]表示运算

$a \leftarrow (b + [a + g(b,c,d) + M[i] + t_i]) \lll s$（仅替换$a$）

t_i是$2^{32} \times ABS(sin(i))$的整数部分

用$round2$依次执行下述16层运算：

a	(b, c, d)	i	s	t
[A	B C D]	1	5	t_{17}]
[D	A B C]	6	9	t_{18}]
[C	D A B]	11	14	t_{19}]
[B	C D A]	0	20	t_{20}]
[A	B C D]	5	5	t_{21}]
[D	A B C]	10	9	t_{22}]
[C	D A B]	15	14	t_{23}]
[B	C D A]	4	20	t_{24}]

a	(b, c, d)	i	s	t
[A	B C D]	9	5	t_{25}]
[D	A B C]	14	9	t_{26}]
[C	D A B]	3	14	t_{27}]
[B	C D A]	8	20	t_{28}]
[A	B C D]	13	5	t_{29}]
[D	A B C]	2	9	t_{30}]
[C	D A B]	7	14	t_{31}]
[B	C D A]	12	20	t_{32}]

7.3 MD5算法描述

- 函数$round3$的结构：用[a b c d i s t]表示运算

$a \leftarrow (b + [a + h(b,c,d) + M[i] + t_i]) \lll s$（仅替换$a$）

t_i是$2^{32} \times ABS(sin(i))$的整数部分

用$round3$依次执行下述16层运算：

a	(b, c, d)	i	s	t
[A	B C D	5	4	t_{33}]
[D	A B C	8	11	t_{34}]
[C	D A B	11	16	t_{35}]
[B	C D A	14	23	t_{36}]
[A	B C D	1	4	t_{37}]
[D	A B C	4	11	t_{38}]
[C	D A B	7	16	t_{39}]
[B	C D A	10	23	t_{40}]

a	(b, c, d)	i	s	t
[A	B C D	13	4	t_{41}]
[D	A B C	0	11	t_{42}]
[C	D A B	3	16	t_{43}]
[B	C D A	6	23	t_{44}]
[A	B C D	9	4	t_{45}]
[D	A B C	12	11	t_{46}]
[C	D A B	15	16	t_{47}]
[B	C D A	2	23	t_{48}]

7.3 MD5算法描述

- 函数$round4$的结构：用[a b c d i s t]表示运算

$a \leftarrow (b + [a + I(b,c,d) + M[i] + t_i]) \lll s$（仅替换$a$）

t_i是$2^{32} \times ABS(sin(i))$的整数部分

用$round4$依次执行下述16层运算：

a	(b, c, d)	i	s	t
[A	B C D]	0	6	t_{49}]
[D	A B C]	7	10	t_{50}]
[C	D A B]	14	15	t_{51}]
[B	C D A]	5	21	t_{52}]
[A	B C D]	12	6	t_{53}]
[D	A B C]	3	10	t_{54}]
[C	D A B]	10	15	t_{55}]
[B	C D A]	1	21	t_{56}]

a	(b, c, d)	i	s	t
[A	B C D]	2	6	t_{57}]
[D	A B C]	15	10	t_{58}]
[C	D A B]	6	15	t_{59}]
[B	C D A]	13	21	t_{60}]
[A	B C D]	4	6	t_{61}]
[D	A B C]	11	10	t_{62}]
[C	D A B]	2	15	t_{63}]
[B	C D A]	9	21	t_{64}]

问题：实现MD5算法。