《现代密码学》第六讲

HASH函数和MAC（一）

上讲内容回顾

- OTP与伪随机数发生器
- 流密码技术的发展及分类
- 移位寄存器与流密码
- RC4算法及其应用
- Estream 算法举例

本讲主要内容

- Hash函数的定义及安全目标
- Hash函数的发展现状
- Hash函数的构造
- 消息鉴别码的定义及安全目标
- 消息鉴别码的发展现状
- 消息鉴别码的构造
- 认证加密模式

本讲主要内容

- Hash函数的定义及安全目标
- Hash函数的发展现状
- Hash函数的构造
- 消息鉴别码的定义及安全目标
- 消息鉴别码的发展现状
- 消息鉴别码的构造
- 认证加密模式

Hash函数定义及安全目标

- Hash函数 **H**: $\{0,1\}^* \rightarrow \{0,1\}^n$又称为**改动检测码** **MDC** (manipulation detection code)，杂凑函数、哈希函数等等。
- 原像空间$\{0,1\}^*$称为消息空间，又可用M表示，消息是任意有限长度；
- 像空间$\{0,1\}^n$称为hash值空间，hash值（散列值、消息摘要）长度固定为n.
- 信息安全作用：完整性

Hash函数定义及安全目标

单向函数定义：

函数 $f:\{0,1\}^* \rightarrow \{0,1\}^*$若满足下列两个条件，则称之为**强单向函数**：

1）计算f(x)是容易的，即f(x)是多项式时间可计算的；

2）计算f函数的逆$f^{-1}(x)$是困难的，即对每一多项式时间概率算法M，每一多项式p(n)和充分大的n($n>n_0$)有

$$\Pr\{M(f(U_n)) \in f^{-1}(f(U_n))\} < 1/p(n)$$

Hash函数定义及安全目标

- 单向性（抗原像）：对于任意给定的消息，计算其哈希值容易．但是，对于给定的哈希值h，要找到M使得H(M)＝h在计算上是不可行的．
- 弱抗碰撞（抗二次原像）：对于给定的消息M_1，要发现另一个消息M_2，满足H(M_1)＝H(M_2)在计算上是不可行的．
- 强抗碰撞：找任意一对不同的消息M_1, M_2 ，使H(M_1)＝H(M_2)在计算上是不可行的．

Hash函数定义及安全目标

- 生日悖论

如果一个房间里有23个或23个以上的人，那么至少有两个人的生日相同的概率要大于50%

- 生日悖论

Th：令$r_1, \ldots, r_n \in \{1,\ldots,N\}$是相互独立的同分布整数，当$n=1.2 \times N^{1/2}$时，

$$\Pr[\exists i \neq j: \ r_i = r_j] \geq \frac{1}{2}$$

Hash函数定义及安全目标

- 生日攻击

令$H:\{0,1\}^* \rightarrow \{0,1\}^n$是一个hash函数.

1. 从$\{0,1\}^*$中任选$2^{n/2}$个元素：$m_1, \ldots, m_{2^{n/2}}$
2. 对 $i = 1,\ldots,2^{n/2}$分别计算$t_i = H(m_i) \in \{0,1\}^n$
3. 寻找碰撞$t_i = t_j$. 若碰撞不存在，则返回第1步重新执行.

根据生日悖论，期望的迭代次数≈2

时间复杂度： $O(2^{n/2})$　　空间复杂度：$(2^{n/2})$

本讲主要内容

- Hash函数的定义及安全目标
- Hash函数的发展现状
- Hash函数的构造
- 消息鉴别码的定义及安全目标
- 消息鉴别码的发展现状
- 消息鉴别码的构造
- 认证加密模式

Hash函数的发展现状

Hash的概念起源于1956年，Dumey用它来解决符号表问题。使得数据表的插入、删除、查询操作可以在平均常数时间完成。

1978年，Merkle和Damagad分别独立设计了MD迭代结构。
1983年，Davies提出一种使用DES构造压缩函数的方法。

1984年，Winternitz首次研究了Davies–Meyer构造的安全性。

Hash函数的发展现状

散列算法MD族是在90年代初由mit laboratory for computer science和RSA data security inc的Ron·Rivest设计的，MD代表消息摘要（message-digest）.
md2、md4和md5都产生一个128位的信息摘要.

- **MD2**
  1989年开发出md2算法. 在这个算法中，首先对信息进行数据补位，使信息的字节长度是16的倍数. 然后，以一个16位的检验和追加到信息末尾。
- **MD4**
  1990年开发出md4算法.
- **MD5**
  1991年，rivest开发出技术上更为趋近成熟的md5算法.
- **RIPEMD-128/160/320**
  RIPEMD由欧洲财团开发和设计.

Hash函数的发展现状

SHA系列算法是NIST 根据Rivest 设计的MD4和MD5开发的算法. 国家安全当局发布SHA作为美国政府标准. SHA表示安全散列算法.

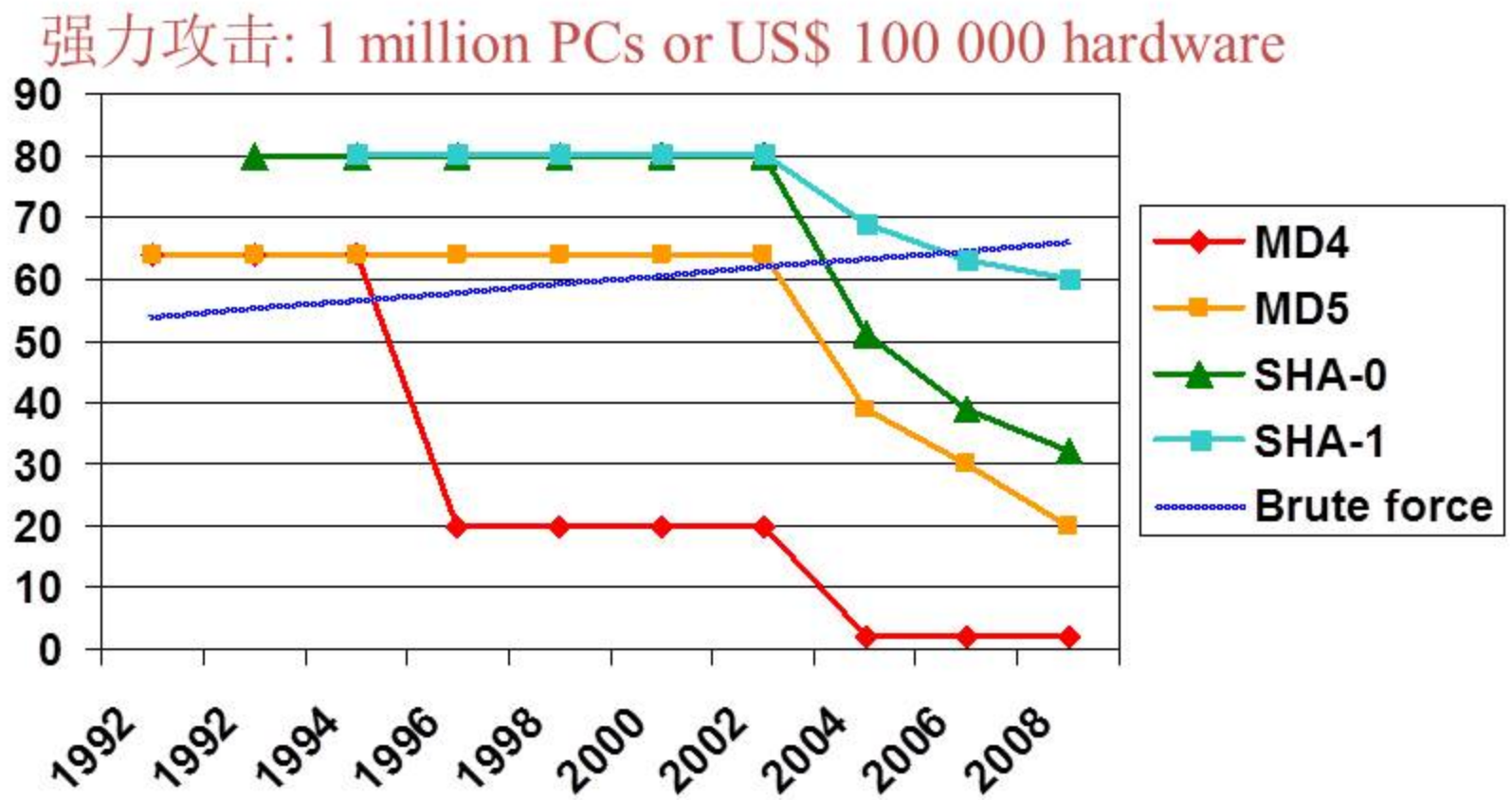
- **SHA-0**
  SHA-0正式地称作SHA，这个版本在发行后不久被指出存在弱点.
- **SHA-1**
  SHA-1是NIST于1994年发布的，它与MD4和MD5散列算法非常相似，被认为是MD4和 MD5的后继者.
- **SHA-2**
  SHA-2实际上分为SHA-224、SHA-256、SHA-384和SHA-512算法.

Hash函数的发展现状

Algorithm	Output size (bits)	Internal state size (bits)	Block size (bits)	Max message size (bits)	Word size (bits)	Rounds	Operations	Collision
SHA-0	160	160	512	$2^{64}-1$	32	80	+,and,or,xor,rotl	Yes
SHA-1	160	160	512	$2^{64}-1$	32	80	+,and,or,xor,rotl	2^{63} attack
SHA-256/224	256/224	256	512	$2^{64}-1$	32	64	+,and,or,xor,shr,rotr	None
SHA-512/384	512/384	512	1024	$2^{128}-1$	64	80	+,and,or,xor,shr,rotr	None

碰撞攻击复杂度

Hash函数的发展现状



碰撞攻击复杂度

Hash函数的发展现状

- NESSIE工程推荐使用的hash算法有SHA-256/384/512和Whirlpool；
- 日本密码研究与评估委员会推荐使用的算法有RIPEMD-160、SHA-256/384/512。
- ECRYPT也在hash算法研究方面举办了一系列活动。
- 此外，NIST于2008年启动新的hash标准的征集活动。
  - 除迭代结构以外的结构
  - 适用于任何平台的压缩函数
- 2008年10月提交文档，收到64个算法，公开56个，51个进入第一轮评估
- 2009年10月，第二轮评估开始，剩余14个算法

Hash函数的发展现状

- 2011年，剩余五个算法

Hash Name	Principal Submitter	Best Attack on Main NIST Requirements	Best Attack on other Hash Requirements
BLAKE	Jean-Philippe Aumasson		
Grøstl	Lars R. Knudsen		
JH	Hongjun Wu	preimage	
Keccak	The Keccak Team		
Skein	Bruce Schneier		

- 2012年10月Keccak 成为SHA-3 standard

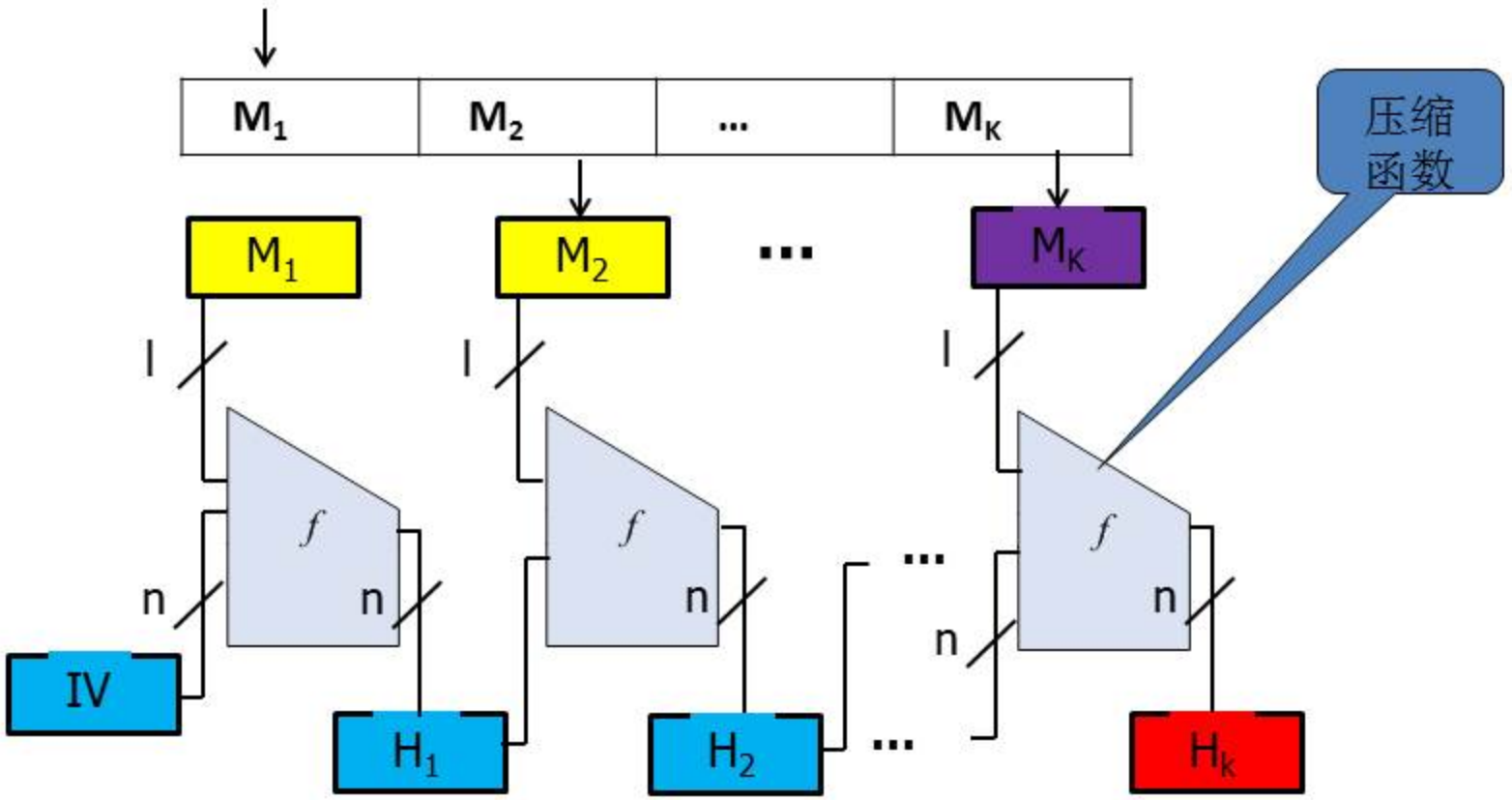
Hash Name	Principal Submitter	Best Attack on Main NIST Requirements	Best Attack on other Hash Requirements
Keccak	The Keccak Team		

GM/T 0004-2012《SM3密码杂凑算法》

本讲主要内容

- Hash函数的定义及安全目标
- Hash函数的发展现状
- Hash函数的构造
- 消息鉴别码的定义及安全目标
- 消息鉴别码的发展现状
- 消息鉴别码的构造
- 认证加密模式

Hash函数的构造



Hash函数的构造

- 固定初始值
- 长消息分块迭代处理
- 长度填充（MD-加固）

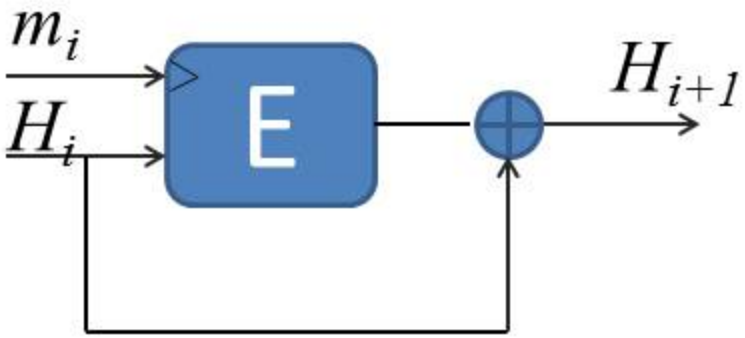
  m||PB, PB=100...00LM，LM为消息长度

Th：如果压缩函数是抗碰撞的，则MD结构构造的hash函数是抗碰撞的

压缩函数既可以由hashing方法构造，也可以由分组密码构造.

Hash函数的构造

- Davies-Meyer压缩函数



Th：如果E是一个理想的密码，则DM压缩函数寻找碰撞的复杂度为生日攻击复杂度

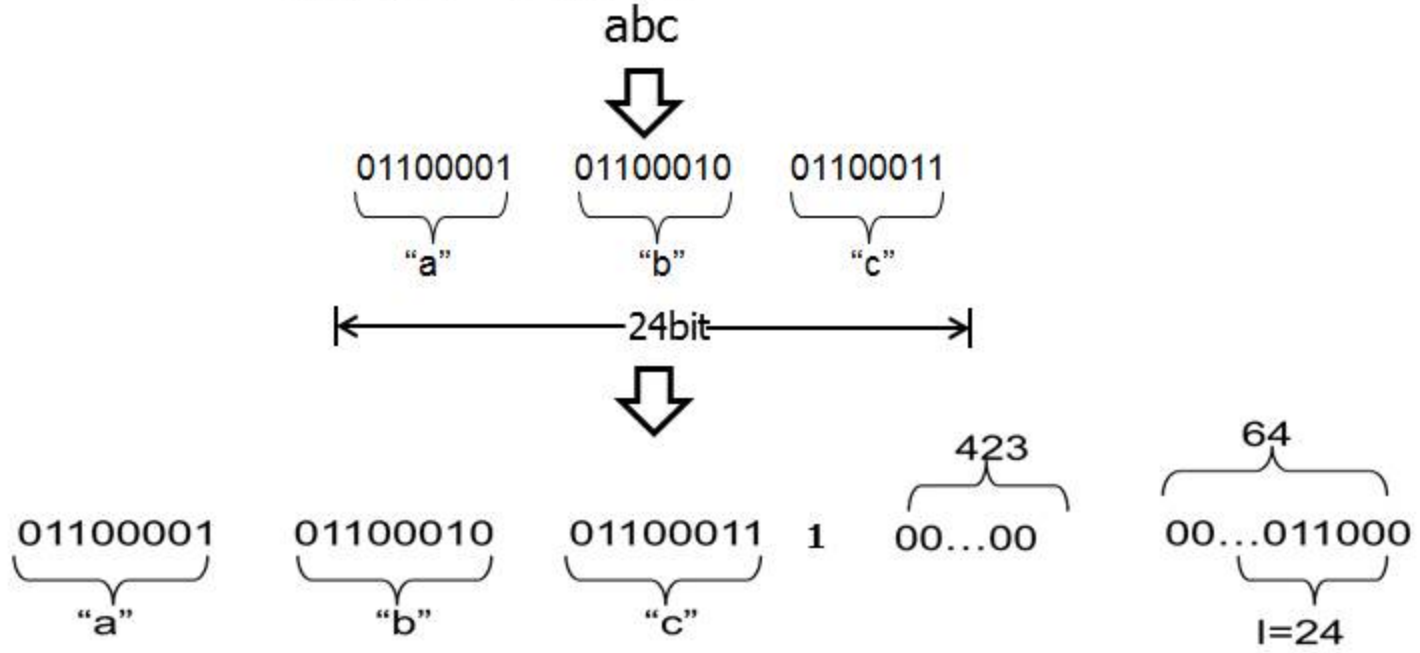
Hash函数的构造

1. SHA-256消息填充

首先将比特“1”添加到消息的末尾，再添加k个零，这里k是方程$l+1+k \equiv 448 \bmod 512$的最小的非负解. 然后再添加一个64比特长的块，其值等于消息M的长度l的二进制表示. 使得填充后的消息的长度为512比特的倍数.

Hash函数的构造

SHA-256消息填充



512=24+5+1+n n=423

Hash函数的构造

2 SHA-256迭代

消息分组和初始值进入MD结构进行迭代压缩

SHA-256的初始变量

$H_0^{(0)}$=6a09e667　　$H_4^{(0)}$=510e527f

$H_1^{(0)}$=bb67ae85　　$H_5^{(0)}$=9b05688c

$H_2^{(0)}$=3c6ef372　　$H_6^{(0)}$=1f83d9ab

$H_3^{(0)}$=a54ff53a　　$H_7^{(0)}$=5be0cd19

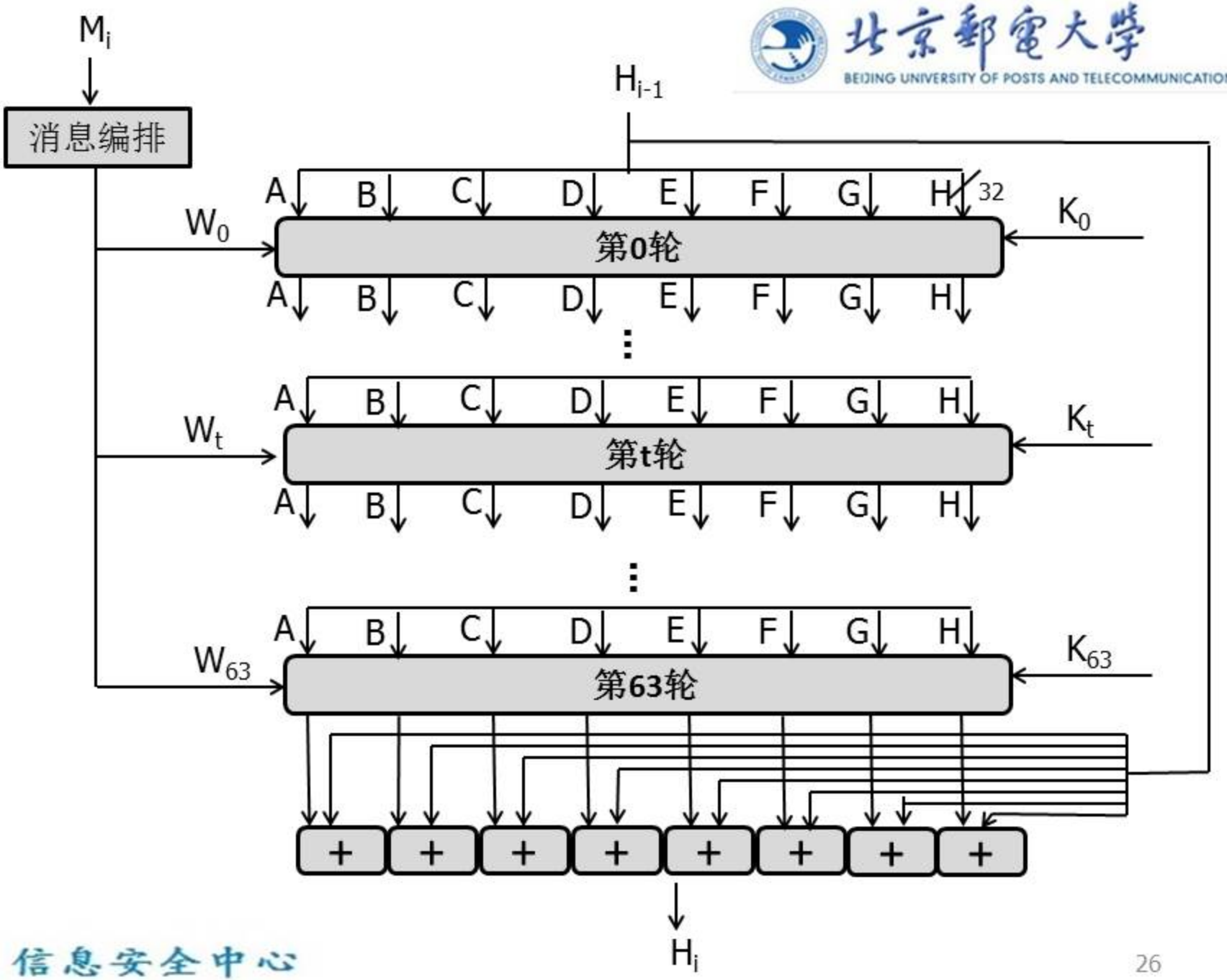
这些初值由计算前8个素数的平方根的小数部分的前32位(二进制)生成

压缩函数的消息分组长度为512比特，压缩函数共64步变换.

Hash函数的构造

SHA-256 压缩函数

- 输入链接变量a、b、c、d、e、f、g和h:
- for t=0 to 63，执行步函数
- 与输入链接变量的副本模加，计算第i个Hash值$H^{(i)}$:


M_i
消息编排
H_{i-1}
A B C D E F G H
32
W_0
K_0
第0轮
A B C D E F G H
W_t
K_t
第t轮
A B C D E F G H
W_{63}
K_{63}
第63轮
+ + + + + + + +
H_i

Hash函数的构造

SHA-256 步函数

- 输入寄存器a、b、c、d、e、f、g和h的当前值
- 输入扩展消息字Wi
- 输入常数Ki
- 更新寄存器

$$T_1 = h + \sum_1^{\{256\}}(e) + Ch(e,f,g) + K_t^{\{256\}} + W_t$$

$$T_2 = \sum_0^{\{256\}}(a) + Maj(a,b,c)$$

$$h = g$$

$$g = f$$

$$f = e$$

$$e = d + T_1$$

$$d = c$$

$$c = b$$

$$b = a$$

$$a = T_1 + T_2$$

$$H_0^{(i)} = a + H_0^{(i-1)} \qquad a = H_0^{(i-1)}$$

$$H_1^{(i)} = b + H_1^{(i-1)} \qquad b = H_1^{(i-1)}$$

$$H_2^{(i)} = c + H_2^{(i-1)} \qquad c = H_2^{(i-1)}$$

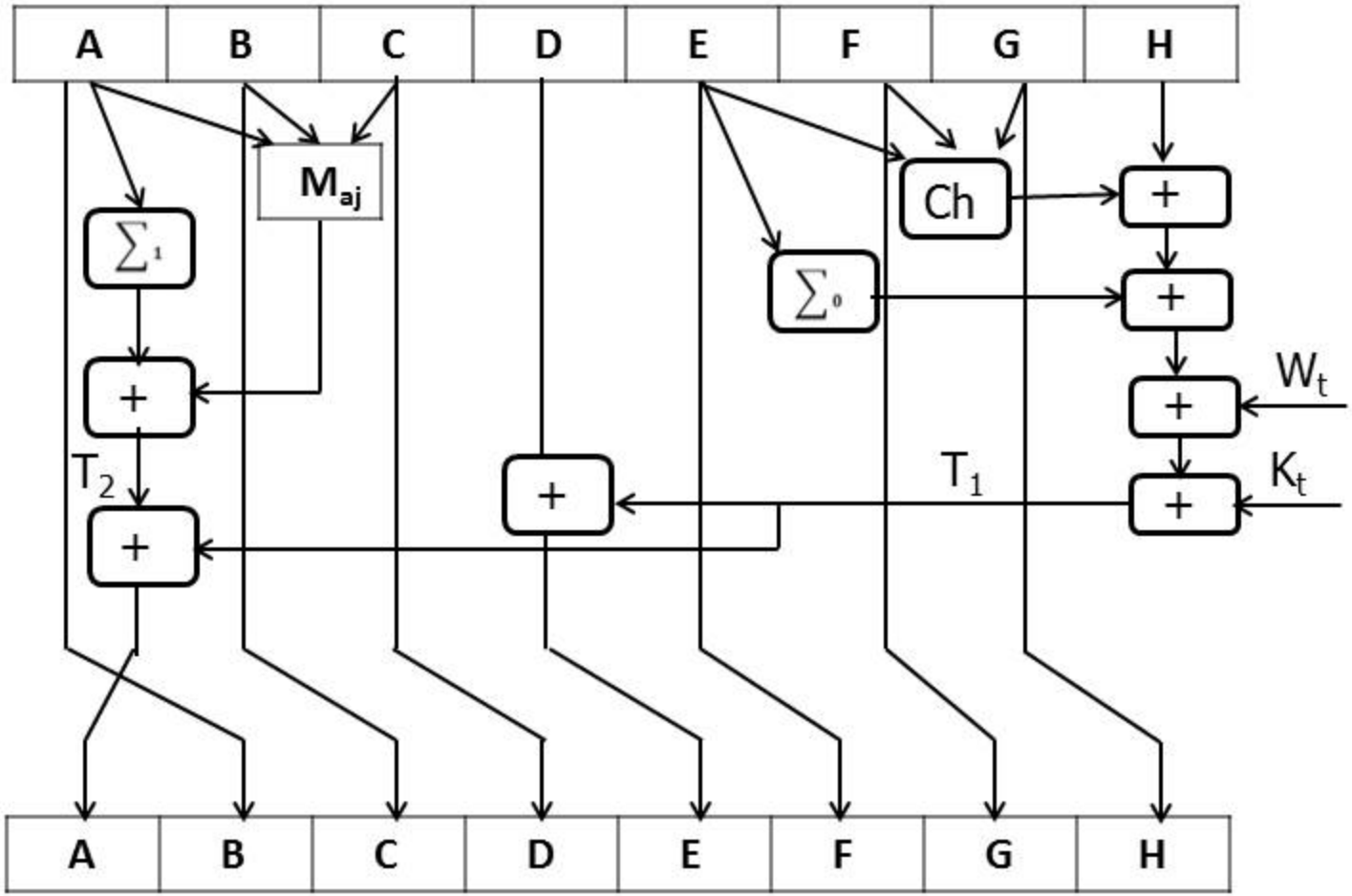
$$H_3^{(i)} = d + H_3^{(i-1)} \qquad d = H_3^{(i-1)}$$

$$H_4^{(i)} = e + H_4^{(i-1)} \qquad e = H_4^{(i-1)}$$

$$H_5^{(i)} = f + H_5^{(i-1)} \qquad f = H_5^{(i-1)}$$

$$H_6^{(i)} = g + H_6^{(i-1)} \qquad g = H_6^{(i-1)}$$

$$H_7^{(i)} = h + H_7^{(i-1)} \qquad h = H_7^{(i-1)}$$



256bit

Hash函数的构造

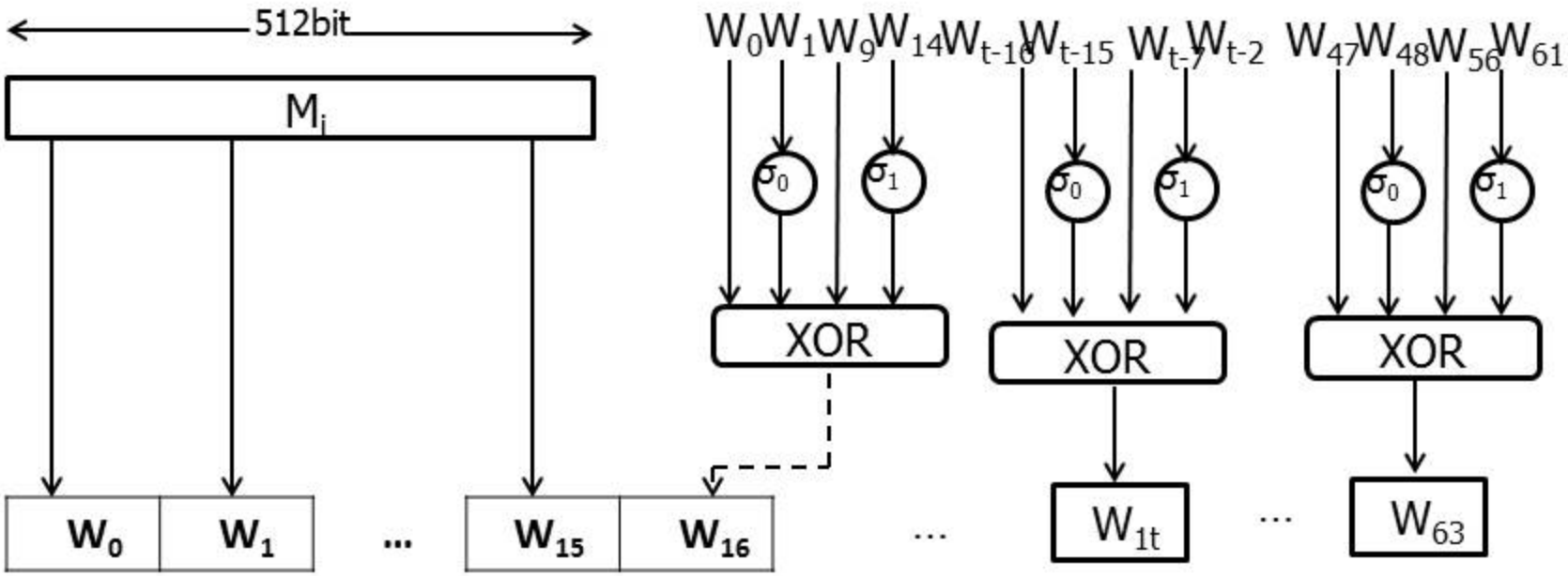
SHA-256的消息编排

当消息填充完成后，将消息块$M^{(1)}$，$M^{(2)}$，...，$M^{(N)}$按序排列，然后执行以下步骤：

- For i=1 to N

$$W_t = \begin{cases} M_t^{(i)} & 0 \le t \le 15 \\ \sigma_1^{256}(W_{t-2}) + W_{t-7} + \sigma_0^{256}(W_{t-15}) + W_{t-16} & 16 \le t \le 63 \end{cases}$$

Hash函数的构造


512bit
M_i
W_0 W_1 ... W_{15} W_{16}
W_0 W_1 W_9 W_{14} W_{t-16} W_{t-15} W_{t-7} W_{t-2} W_{47} W_{48} W_{56} W_{61}
σ_0 σ_1
XOR
W_{1t}
W_{63}

Hash函数的构造

SHA-256使用了6个逻辑函数，设x, y和z为3个32比特长的自变量，输出结果都是32比特长的字，逻辑函数定义如下：

$$Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z)$$

$$Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$$

$$\sum_0^{\{256\}}(x) = ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x)$$

$$\sum_1^{\{256\}}(x) = ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x)$$

$$\sigma_0^{\{256\}}(x) = ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x)$$

$$\sigma_1^{\{256\}}(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x)$$

Hash函数的构造

SHA-256的常数

共使用了64个32位字长的常数，它们分别由最小的64个素数的三次方根的小数部分的前32位产生（二进制表示）

428a2f98 71374491 b5c0fbcf e9b5dba5
3956c25b 59f111f1 923f82a4 ab1c5ed5
d807aa98 12835b01 243185be 550c7dc3
72be5d74 80deb1fe 9bdc06a7 c19bf174
e49b69c1 efbe4786 0fc19dc6 240ca1cc
2de92c6f 4a7484aa 5cb0a9dc 76f988da
983e5152 a831c66d b00327c8 bf597fc7
c6e00bf3 d5a79147 06ca6351 14292967
27b70a85 2e1b2138 4d2c6dfc 53380d13
650a7354 766a0abb 81c2c92e 92722c85
a2bfe8a1 a81a664b c24b8b70 c76c51a3
d192e819 d6990624 f40e3585 106aa070
19a4c116 1e376c08 2748774c 34b0bcb5
391c0cb3 4ed8aa4a 5b9cca4f 682e6ff3
748f82ee 78a5636f 84c87814 8cc70208
90befffa a4506ceb bef9a3f7 c67178f2

Hash函数的构造

3 最后一轮迭代输出的链接变量值，即为散列值，长度为256比特.

SHA-224算法可用来Hash任意l(0≤l≤2^{64})位长的消息M。算法的具体实现和SHA-256基本一样，不同的是以下两点：

1) 初始Hash值H(0)的设置不同；

2) 输出224比特长的消息摘要，即剪切H(N)的左边224比特位产生

Hash函数的构造

Published in	Year	Attack method	Attack	Variant	Rounds	Complexity
New Collision Attacks Against Up To 24-step SHA-2[29]	2008	Deterministic	Collision	SHA-256	24/64	$2^{28.5}$
				SHA-512	24/80	$2^{32.5}$
Preimages for step-reduced SHA-2[30]	2009	Meet-in-the-middle	Preimage	SHA-256	42/64	$2^{251.7}$
					43/64	$2^{254.9}$
				SHA-512	42/80	$2^{502.3}$
					46/80	$2^{511.5}$
Advanced meet-in-the-middle preimage attacks[31]	2010	Meet-in-the-middle	Preimage	SHA-256	42/64	$2^{248.4}$
				SHA-512	42/80	$2^{494.6}$
Higher-Order Differential Attack on Reduced SHA-256[2]	2011	Differential	Pseudo-collision	SHA-256	46/64	2^{178}
					46/64	2^{46}
Bicliques for Preimages: Attacks on Skein-512 and the SHA-2 family[1]	2011	Biclique	Preimage	SHA-256	45/64	$2^{255.5}$
				SHA-512	50/80	$2^{511.5}$
			Pseudo-preimage	SHA-256	52/64	2^{255}
				SHA-512	57/80	2^{511}
Improving Local Collisions: New Attacks on Reduced SHA-256[32]	2013	Differential	Collision	SHA-256	31/64	$2^{65.5}$
			Pseudo-collision	SHA-256	38/64	2^{37}
Branching Heuristics in Differential Collision Search with Applications to SHA-512[33]	2014	Heuristic differential	Pseudo-collision	SHA-512	38/80	$2^{40.5}$

本节要点小结

- Hash函数的定义及安全目标
- Hash函数的发展现状
- Hash函数的构造

THE END!