

密码学原理与实践 (第三版)

第4章 Hash函数



苏 明

[加] Douglas R. Stinson 著

冯登国 等译



概览

- **4. 1 Hash函数与数据完整性**
- **4. 2 Hash函数的安全性**
 - 4. 2. 1 随机预言机模型
 - 4. 2. 2 随机预言机中的算法
 - 4. 2. 3 安全性准则的比较
- **4. 3 迭代Hash函数**
 - 4. 3. 1 Merkle-Damgard 结构
 - 4. 3. 2 安全Hash算法
- **4. 4 消息认证码**
- **4. 5 无条件安全消息认证码**



Hash函数与数据完整性

- Hash

- 密码学的Hash函数可以保障数据的完整性
产生 *指纹* $y=H(x)$
消息摘要 (Message Digest)



Hash函数与数据完整性

■ Hash函数、带密钥的Hash族（MAC）

定义 4.1 一个 Hash 族是满足下列条件的四元组 $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$:

1. \mathcal{X} 是所有可能的消息的集合。
 2. \mathcal{Y} 是由所有可能的消息摘要或认证标签构成的有限集。
 3. \mathcal{K} 是密钥空间, 是所有可能的密钥构成的有限集。
 4. 对每个 $K \in \mathcal{K}$, 存在一个 Hash 函数 $h_K \in \mathcal{H}$, $h_K : \mathcal{X} \rightarrow \mathcal{Y}$ 。
-

令 $\mathcal{F}^{\mathcal{X}, \mathcal{Y}}$ 为所有从 \mathcal{X} 到 \mathcal{Y} 的函数集合。假定 $|\mathcal{X}| = N$ 和 $|\mathcal{Y}| = M$ 。则显然 $|\mathcal{F}^{\mathcal{X}, \mathcal{Y}}| = M^N$



Hash函数的安全性

通常 $N \gg M$, 碰撞永远存在。关键是：难于找到
如下问题是难解的

问题 4.1 原像 (Preimage)

实例：Hash 函数 $h: \mathcal{X} \rightarrow \mathcal{Y}$ 和 $y \in \mathcal{Y}$ 。

找出： $x \in \mathcal{X}$ 使得 $h(x) = y$ 。

问题 4.2 第二原像

实例：Hash 函数 $h: \mathcal{X} \rightarrow \mathcal{Y}$ 和 $x \in \mathcal{X}$ 。

找出： $x' \in \mathcal{X}$ 使得 $x' \neq x$ ，并且 $h(x') = h(x)$ 。

问题 4.3 碰撞

实例：Hash 函数 $h: \mathcal{X} \rightarrow \mathcal{Y}$ 。

找出： $x, x' \in \mathcal{X}$ 使得 $x' \neq x$ ，并且 $h(x') = h(x)$ 。



随机预言机模型

■ The Random Oracle Model

--looking up the value $h(x)$ in a giant book of random numbers

若函数设计的好，求出函数 h 在点 x 的值应该是得到 $h(x)$ 的唯一有效方法



随机预言机模型

Example

假定 Hash 函数 $h: \mathbb{Z}_n \times \mathbb{Z}_n \rightarrow \mathbb{Z}_n$ 是一个线性函数, 令

$$h(x, y) = ax + by \bmod n \quad a, b \in \mathbb{Z}_n \text{ 且 } n \geq 2 \text{ 是正整数。}$$

这不是一个好的设计!

$$h(rx_1 + sx_2 \bmod n, ry_1 + sy_2 \bmod n) = rh(x_1, y_1) + sh(x_2, y_2) \bmod n$$



随机预言机模型

- 通俗的说起来，没有查询过的Hash值是
均匀分布的

Assumption

假定 $h \in \mathcal{F}^{\mathcal{X}, \mathcal{Y}}$ 是随机选择的，令 $\mathcal{X}_0 \subseteq \mathcal{X}$ 。假定当且仅当 $x \in \mathcal{X}_0$ 时， $h(x)$ (通过查询 h 的谕示器) 被确定。则对所有的 $x \in \mathcal{X} \setminus \mathcal{X}_0$ 和 $y \in \mathcal{Y}$ ，都有 $\Pr[h(x) = y] = 1/M$ 。



随机预言机模型

■ (ϵ, Q)

Success Probability ϵ (worst case/average case)

Queries Q

■ Algorithms

算法 4.1 Find-Preimage(h, y, Q)

选择任意的 $\mathcal{X}_0 \subseteq \mathcal{X}$, $|\mathcal{X}_0| = Q$

for each $x \in \mathcal{X}_0$

do $\left\{ \begin{array}{l} \text{if } h(x) = y \\ \text{then return } (x) \end{array} \right.$

return (failure)



随机预言机模型

算法4.1的平均情况的成功率为 $\epsilon = 1 - (1 - 1/M)^Q$

Proof. 令 E_i 表示事件 “ $h(x_i) = y$ ”.

$$\begin{aligned}\Pr[E_1 \vee E_2 \vee \cdots \vee E_Q] &= 1 - \Pr[E_1^c \wedge E_2^c \wedge \cdots \wedge E_Q^c] \\ &= 1 - \left(1 - \frac{1}{M}\right)^Q,\end{aligned}$$

因此, 对所有的 $y \in \mathcal{Y}$ 取平均既得结论。

如果 $Q \ll M$, 那么 $\epsilon \approx Q/M$



随机预言机模型

算法 4.2 Find-Second-Preimage(h, x, Q)

$y \leftarrow h(x)$

选择 $\mathcal{X}_0 \subseteq \mathcal{X} \setminus \{x\}$, $|\mathcal{X}_0| = Q - 1$

for each $x_0 \in \mathcal{X}_0$

do $\begin{cases} \text{if } h(x_0) = y \\ \text{then return } (x_0) \end{cases}$

return (failure)

类似的, 算法 4.2 的成功率是 $\epsilon = 1 - (1 - 1/M)^{Q-1}$



随机预言机模型

- 针对碰撞问题，直接寻找算法如下

算法 4.3 Find-Collision(h, Q)

选择 $\mathcal{X}_0 \subseteq \mathcal{X}$, $|\mathcal{X}_0| = Q$

for each $x \in \mathcal{X}_0$

do $y_x \leftarrow h(x)$

if 对某一 $x' \neq x$, 有 $y_x = y_{x'}$

then return (x, x')

else return (failure)



碰撞问题

■ 生日悖论

定理 4.4 对任意的 $\mathcal{X}_0 \subseteq \mathcal{X}$ ，且 $|\mathcal{X}_0| = Q$ ，算法 4.3 的成功率为

$$\epsilon = 1 - \left(\frac{M-1}{M} \right) \left(\frac{M-2}{M} \right) \cdots \left(\frac{M-Q+1}{M} \right)$$



碰撞问题

考虑到 $\lim_{x \rightarrow 0} \frac{e^x}{x+1} = 1$

级数展开表达式 $e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} \dots$

$$\prod_{i=1}^{Q-1} \left(1 - \frac{i}{M}\right) \approx e^{-\frac{Q(Q-1)}{2M}}$$

可以知道 $Q^2 \approx 2M \ln\left(\frac{1}{1-\epsilon}\right)$

如果 $\epsilon = 0.5$, 那么 $Q \approx 1.17\sqrt{M}$



迭代Hash函数

- Hash函数 本质上是压缩函数
- 延拓：将一个compress压缩函数扩展为无限定义域的Hash函数
（支持任意长度的消息输入）
- 迭代



迭代Hash函数

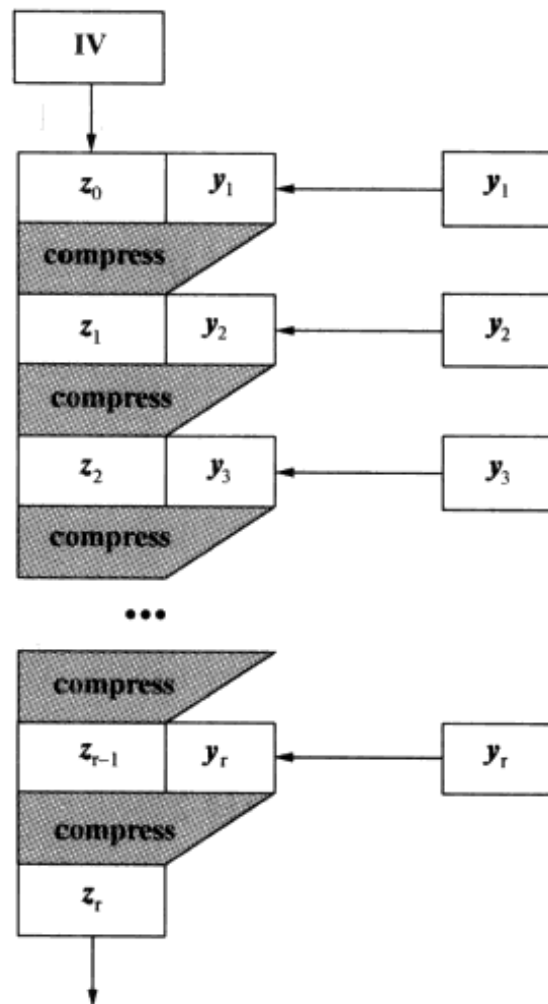
设 $\text{compress}: \{0, 1\}^{m+t} \rightarrow \{0, 1\}^m$ 是一个压缩函数 (这里 $t \geq 1$)

- 构造一个迭代Hash函数

$$h: \bigcup_{i=m+t+1}^{\infty} \{0, 1\}^i \rightarrow \{0, 1\}^{\ell}$$

迭代Hash函数

Overview



迭代Hash函数

■ 预处理（通过填充，保证 $x \mapsto y$ 是单射）

给定一个输入比特串 x ，其中 $|x| \geq m + t + 1$



$$y = y_1 \parallel y_2 \parallel \cdots \parallel y_r \quad \text{其中对 } 1 \leq i \leq r, \text{ 有 } |y_i| = t。$$

■ 处理

$$\begin{aligned} z_0 &\leftarrow \text{IV} \\ z_1 &\leftarrow \text{compress}(z_0 \parallel y_1) \\ z_2 &\leftarrow \text{compress}(z_1 \parallel y_2) \\ &\vdots \\ z_r &\leftarrow \text{compress}(z_{r-1} \parallel y_r) \end{aligned}$$

■ 输出变换（可选）

设 $g: \{0, 1\}^m \rightarrow \{0, 1\}^\ell$ 是一个公开函数。定义 $h(x) = g(z_r)$



迭代Hash函数

- 大多数Hash函数采用了迭代结构
- SHA1...



Merkle-Damgård 结构

假定 $\text{compress}: \{0, 1\}^{m+t} \rightarrow \{0, 1\}^m$ 是一个碰撞稳固压缩函数, 其中 $t \geq 1$



构造一个碰撞稳固 Hash 函数 $h: \mathcal{X} \rightarrow \{0, 1\}^m$

$$\mathcal{X} = \bigcup_{i=m+t+1}^{\infty} \{0, 1\}^i$$



Merkle-Damgård 结构

构造

- 预处理

假定 $|x| = n \geq m + t + 1$

$$x = x_1 \parallel x_2 \parallel \cdots \parallel x_k$$

其中

$$|x_1| = |x_2| = \cdots = |x_{k-1}| = t - 1$$

$$|x_k| = t - 1 - d$$

$$k = \left\lceil \frac{n}{t-1} \right\rceil$$

Merkle-Damgård 结构

构造

■ 预处理

$$y(x) = y_1 \parallel y_2 \parallel \cdots \parallel \boxed{y_k} \parallel \boxed{y_{k+1}}$$

$\boxed{y_k}$ x_k 的右边添加 d 个 0

$\boxed{y_{k+1}}$ 在左边添加 0, 使得 $|y_{k+1}| = t - 1$

$x \mapsto y(x)$ 为单射

Merkle-Damgård 结构

精巧的设计

算法 Merkle-Damgård(x)

external compress

注释 **compress**: $\{0, 1\}^{m+t} \rightarrow \{0, 1\}^m$, 其中 $t \geq 2$

$n \leftarrow |x|$

$k \leftarrow \lceil n/(t-1) \rceil$

$d \leftarrow n - k(t-1)$

for $i \leftarrow 1$ **to** $k-1$

do $y_i \leftarrow x_i$

$y_k \leftarrow x_k \parallel 0^d$

$y_{k+1} \leftarrow d$ 的二进制表示

$z_1 \leftarrow \underline{0^{m+1}} \parallel y_1$

$g_1 \leftarrow \text{compress}(z_1)$

for $i \leftarrow 1$ **to** k

do $\begin{cases} z_{i+1} \leftarrow g_i \parallel \underline{1} \parallel y_{i+1} \\ g_{i+1} \leftarrow \text{compress}(z_{i+1}) \end{cases}$

$h(x) \leftarrow g_{k+1}$

return ($h(x)$)

Merkle-Damgård 结构

compress: $\{0, 1\}^{m+t} \rightarrow \{0, 1\}^m$ 是一个碰撞稳固的压缩函数, 其中 $t \geq 2$

h: $\bigcup_{i=m+t+1}^{\infty} \{0, 1\}^i \rightarrow \{0, 1\}^m$ 是一个碰撞稳固的 Hash 函数。

算法 Merkle-Damgård(x)

external compress

注释 **compress:** $\{0, 1\}^{m+t} \rightarrow \{0, 1\}^m$, 其中 $t \geq 2$

$n \leftarrow |x|$

$k \leftarrow \lceil n/(t-1) \rceil$

$d \leftarrow n - k(t-1)$

for $i \leftarrow 1$ **to** $k-1$

do $y_i \leftarrow x_i$

$y_k \leftarrow x_k \parallel 0^d$

$y_{k+1} \leftarrow d$ 的二进制表示

$z_1 \leftarrow 0^{m+1} \parallel y_1$

$g_1 \leftarrow \text{compress}(z_1)$

for $i \leftarrow 1$ **to** k

do $\begin{cases} z_{i+1} \leftarrow g_i \parallel 1 \parallel y_{i+1} \\ g_{i+1} \leftarrow \text{compress}(z_{i+1}) \end{cases}$

$h(x) \leftarrow g_{k+1}$

return ($h(x)$)

Proof:

假定可以找到 $x \neq x'$ 使得

$$h(x) = h(x')$$

我们将在多项式时间内找到
compress 的碰撞。

预处理后

$$y(x) = y_1 \parallel y_2 \parallel \cdots \parallel y_{k+1}$$

$$y(x') = y'_1 \parallel y'_2 \parallel \cdots \parallel y'_{\ell+1}$$

Merkle-Damgård 结构

compress: $\{0, 1\}^{m+t} \rightarrow \{0, 1\}^m$ 是一个碰撞稳固的压缩函数, 其中 $t \geq 2$

h: $\bigcup_{i=m+t+1}^{\infty} \{0, 1\}^i \rightarrow \{0, 1\}^m$ 是一个碰撞稳固的 Hash 函数。

算法 Merkle-Damgård(x)

external compress

注释 **compress:** $\{0, 1\}^{m+t} \rightarrow \{0, 1\}^m$, 其中 $t \geq 2$

$n \leftarrow |x|$

$k \leftarrow \lceil n/(t-1) \rceil$

$d \leftarrow n - k(t-1)$

for $i \leftarrow 1$ **to** $k-1$

do $y_i \leftarrow x_i$

$y_k \leftarrow x_k \parallel 0^d$

$y_{k+1} \leftarrow d$ 的二进制表示

$z_1 \leftarrow 0^{m+1} \parallel y_1$

$g_1 \leftarrow \text{compress}(z_1)$

for $i \leftarrow 1$ **to** k

do $\begin{cases} z_{i+1} \leftarrow g_i \parallel 1 \parallel y_{i+1} \\ g_{i+1} \leftarrow \text{compress}(z_{i+1}) \end{cases}$

$h(x) \leftarrow g_{k+1}$

return ($h(x)$)

Proof continued:

情况 1: $|x| \not\equiv |x'| \pmod{t-1}$

$d \neq d'$ 且 $y_{k+1} \neq y'_{\ell+1}$ (最后一个块填充的长度信息)

因此我们有

$$\text{compress}(g_k \parallel 1 \parallel y_{k+1}) = g_{k+1}$$

$$= g'_{\ell+1}$$

$$= \text{compress}(g'_{\ell} \parallel 1 \parallel y'_{\ell+1})$$

于是我们找到了compress的一个碰撞

Merkle-Damgård 结构

compress: $\{0,1\}^{m+t} \rightarrow \{0,1\}^m$ 是一个碰撞稳固的压缩函数, 其中 $t \geq 2$

h: $\bigcup_{i=m+t+1}^{\infty} \{0,1\}^i \rightarrow \{0,1\}^m$ 是一个碰撞稳固的 Hash 函数。

算法 Merkle-Damgård(x)

external compress

注释 **compress:** $\{0,1\}^{m+t} \rightarrow \{0,1\}^m$, 其中 $t \geq 2$

$n \leftarrow |x|$

$k \leftarrow \lceil n/(t-1) \rceil$

$d \leftarrow n - k(t-1)$

for $i \leftarrow 1$ **to** $k-1$

do $y_i \leftarrow x_i$

$y_k \leftarrow x_k \parallel 0^d$

$y_{k+1} \leftarrow d$ 的二进制表示

$z_1 \leftarrow 0^{m+1} \parallel y_1$

$g_1 \leftarrow \text{compress}(z_1)$

for $i \leftarrow 1$ **to** k

do $\begin{cases} z_{i+1} \leftarrow g_i \parallel 1 \parallel y_{i+1} \\ g_{i+1} \leftarrow \text{compress}(z_{i+1}) \end{cases}$

$h(x) \leftarrow g_{k+1}$

return ($h(x)$)

Proof continued:

情况 2: $|x| \equiv |x'| \pmod{t-1}$

情况 2a: $|x| = |x'|$

此时有 $k = \ell$ 和 $y_{k+1} = y'_{k+1}$ 于是

$$\text{compress}(g_k \parallel 1 \parallel y_{k+1}) = g_{k+1} = g'_{k+1} = \text{compress}(g'_k \parallel 1 \parallel y'_{k+1})$$

如果 $g_k \neq g'_k$, 则找到了 **compress** 的碰撞

假定 $g_k = g'_k$ 。则有

$$\text{compress}(g_{k-1} \parallel 1 \parallel y_k) = g_k = g'_k = \text{compress}(g'_{k-1} \parallel 1 \parallel y'_k)$$

假定没有找到碰撞, 重复下去...

$$\text{compress}(0^{m+1} \parallel y_1) = g_1 = g'_1 = \text{compress}(0^{m+1} \parallel y'_1)$$

假定 $y_1 = y'_1$ 。这样对 $1 \leq i \leq k+1$ 都有 $y_i = y'_i$

因为映射 $x \mapsto y(x)$ 是单射, & $x \neq x'$. 矛盾。 26

Merkle-Damgård 结构

$\text{compress}: \{0, 1\}^{m+t} \rightarrow \{0, 1\}^m$ 是一个碰撞稳固的压缩函数, 其中 $t \geq 2$

$h: \bigcup_{i=m+t+1}^{\infty} \{0, 1\}^i \rightarrow \{0, 1\}^m$ 是一个碰撞稳固的 Hash 函数。

算法 Merkle-Damgård(x)

external compress

注释 $\text{compress}: \{0, 1\}^{m+t} \rightarrow \{0, 1\}^m$, 其中 $t \geq 2$

$n \leftarrow |x|$

$k \leftarrow \lceil n/(t-1) \rceil$

$d \leftarrow n - k(t-1)$

for $i \leftarrow 1$ **to** $k-1$

do $y_i \leftarrow x_i$

$y_k \leftarrow x_k \parallel 0^d$

$y_{k+1} \leftarrow d$ 的二进制表示

$z_1 \leftarrow 0^{m+1} \parallel y_1$

$g_1 \leftarrow \text{compress}(z_1)$

for $i \leftarrow 1$ **to** k

do $\begin{cases} z_{i+1} \leftarrow g_i \parallel 1 \parallel y_{i+1} \\ g_{i+1} \leftarrow \text{compress}(z_{i+1}) \end{cases}$

$h(x) \leftarrow g_{k+1}$

return ($h(x)$)

Proof continued:

情况 2b: $|x| \neq |x'|$

设 $|x'| > |x|$, 因此 $\ell > k$

假定没有找到 compress 的碰撞, 最后总有

$$\text{compress}(0^{m+1} \parallel y_1) = g_1 = g'_{\ell-k+1} = \text{compress}(g'_{\ell-k} \parallel 1 \parallel y'_{\ell-k+1})$$

因为第 $m+1$ 个比特不同, 于是我们找到了 compress 的一个碰撞。



Merkle-Damgård 结构

■ **t=1**

假定 $|x| = n \geq m + 2$ 。用一种特殊方式对 x 编码。

$$f(0) = 0$$

$$f(1) = 01$$

编码 $x \mapsto y = y(x)$ 满足

1. $x \rightarrow y(x)$ 是单射
2. $y(x)$: 没有任何编码是别的编码的后缀 (*Postfix-free*)



Merkle-Damgård 结构

Algorithm MERKLE-DAMGÅRD2(x)

external compress

comment: $\text{compress} : \{0, 1\}^{m+1} \rightarrow \{0, 1\}^m$

$n \leftarrow |x|$

$y \leftarrow \underline{11} \parallel f(x_1) \parallel f(x_2) \parallel \cdots \parallel f(x_n)$

denote $y = y_1 \parallel y_2 \parallel \cdots \parallel y_k$, where $y_i \in \{0, 1\}^m, 1 \leq i \leq k$

$g_1 \leftarrow \text{compress}(0^m \parallel y_1)$

for $i \leftarrow 1$ **to** $k - 1$

do $g_{i+1} \leftarrow \text{compress}(g_i \parallel y_{i+1})$

return (g_k)



Merkle-Damgård 结构

Algorithm MERKLE-DAMGÅRD2(x)

external compress

comment: $\text{compress} : \{0, 1\}^{m+1} \rightarrow \{0, 1\}^m$

$n \leftarrow |x|$

$y \leftarrow \underline{11} \parallel f(x_1) \parallel f(x_2) \parallel \cdots \parallel f(x_n)$

denote $y = y_1 \parallel y_2 \parallel \cdots \parallel y_k$, where $y_i \in \{0, 1\}^m, 1 \leq i \leq k$

$g_1 \leftarrow \text{compress}(0^m \parallel y_1)$

for $i \leftarrow 1$ **to** $k - 1$

do $g_{i+1} \leftarrow \text{compress}(g_i \parallel y_{i+1})$

return (g_k)

编码 $x \mapsto y = y(x)$ 满足

1. $x \rightarrow y(x)$ 是单射
2. $y(x)$: 没有任何编码是别的编码的后缀(Postfix-free)

Merkle-Damgård 结构

假定 $\text{compress}: \{0, 1\}^{m+1} \rightarrow \{0, 1\}^m$ 是一个碰撞稳固的压缩函数。

$$h: \bigcup_{i=m+2}^{\infty} \{0, 1\}^i \rightarrow \{0, 1\}^m$$

是一个碰撞稳固的 Hash 函数。

Algorithm MERKLE-DAMGÅRD2(x)

external compress

comment: $\text{compress}: \{0, 1\}^{m+1} \rightarrow \{0, 1\}^m$

$n \leftarrow |x|$

$y \leftarrow \text{11} \parallel f(x_1) \parallel f(x_2) \parallel \cdots \parallel f(x_n)$

denote $y = y_1 \parallel y_2 \parallel \cdots \parallel y_k$, where $y_i \in \{0, 1\}^m, 1 \leq i \leq k$

$g_1 \leftarrow \text{compress}(0^m \parallel y_1)$

for $i \leftarrow 1$ **to** $k - 1$

do $g_{i+1} \leftarrow \text{compress}(g_i \parallel y_{i+1})$

return (g_k)

Proof:

假定我们可找到 $x \neq x'$ 使得 $h(x) = h(x')$ 。令

$$y(x) = y_1 y_2 \cdots y_k \quad y(x') = y'_1 y'_2 \cdots y'_\ell$$

情况 1: $k = \ell$ 。

或者找到 compress 的一个碰撞

或者得到 $y = y'$

意味着 $x = x'$ ，矛盾。

Merkle-Damgård 结构

假定 $\text{compress}: \{0, 1\}^{m+1} \rightarrow \{0, 1\}^m$ 是一个碰撞稳固的压缩函数。

$$h: \bigcup_{i=m+2}^{\infty} \{0, 1\}^i \rightarrow \{0, 1\}^m$$

是一个碰撞稳固的 Hash 函数。

Algorithm MERKLE-DAMGÅRD2(x)

external compress

comment: $\text{compress}: \{0, 1\}^{m+1} \rightarrow \{0, 1\}^m$

$n \leftarrow |x|$

$y \leftarrow \text{11} \parallel f(x_1) \parallel f(x_2) \parallel \cdots \parallel f(x_n)$

denote $y = y_1 \parallel y_2 \parallel \cdots \parallel y_k$, where $y_i \in \{0, 1\}^m, 1 \leq i \leq k$

$g_1 \leftarrow \text{compress}(0^m \parallel y_1)$

for $i \leftarrow 1$ to $k - 1$

do $g_{i+1} \leftarrow \text{compress}(g_i \parallel y_{i+1})$

return (g_k)

Proof:

情况 2: $k \neq \ell$

设 $\ell > k$

假定没有找到 compress 的碰撞, 则

$$\begin{aligned} y_k &= y'_\ell \\ y_{k-1} &= y'_{\ell-1} \\ &\vdots \\ y_1 &= y'_{\ell-k+1} \end{aligned}$$

这恰好与“无后缀”(Postfix-free)性质矛盾。

Merkle-Damgård 结构

■ 可扩展的Hash函数结构

定理 假定 $\text{compress}: \{0, 1\}^{m+t} \rightarrow \{0, 1\}^m$ 是一个碰撞稳固的压缩函数, 其中 $t \geq 1$ 。则存在一个碰撞稳固的 Hash 函数

$$h: \bigcup_{i=m+t+1}^{\infty} \{0, 1\}^i \rightarrow \{0, 1\}^m$$

在求 h 的值的时候, compress 最多被计算的次数为

$$\begin{array}{ll} 1 + \left\lceil \frac{n}{t-1} \right\rceil & \text{如果 } t \geq 2 \\ 2n + 2 & \text{如果 } t = 1 \end{array}$$

其中 $|x| = n$ 。



安全Hash算法

SHA-1

- 分组512比特, word(32 比特) based
- 基本运算: 位运算

$X \wedge Y$

X 和 Y 的逻辑“和”

$X \vee Y$

X 和 Y 的逻辑“或”

$X \oplus Y$

X 和 Y 的逻辑“异或”

$\neg X$

X 的逻辑“补”

$X + Y$

模 2^{32} 整数加

$\text{ROTL}^s(X)$

X 循环左移 s 个位置 ($0 \leq s \leq 31$)



安全Hash算法

■ 填充方案

算法 SHA-1-PAD(x)

注释 $|x| \leq 2^{64} - 1$

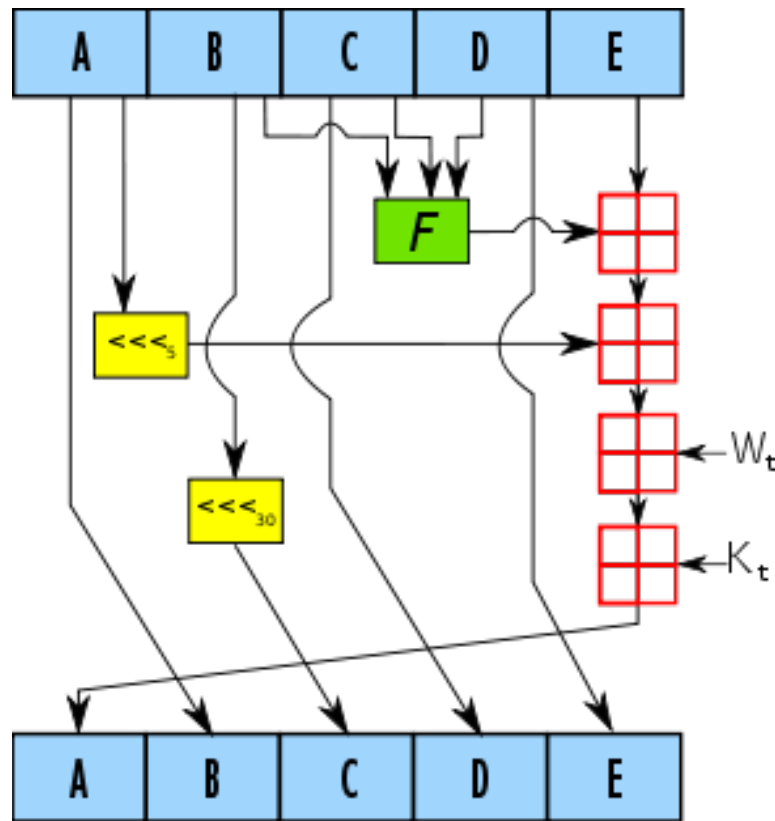
$d \leftarrow (447 - |x|) \bmod 512$

$\ell \leftarrow |x|$ 的二进制表示, 其中 $|\ell| = 64$

$y \leftarrow x \parallel 1 \parallel 0^d \parallel \ell$

于是 y 可以按照512比特分组 $y = M_1 \parallel M_2 \parallel \cdots \parallel M_n$

SHA-1 Overview



SHA-1 压缩算法中的一个循环。A, B, C, D 和 E 是这个state中的 32 位元文字； F 是会变化的非线性函数； \lll_n 代表bit向左循环移动 n 个位置。 n 因操作而异。田代表modulo 2^{32} 之下的加法， K_t 是一个常数。



安全Hash算法

■ 压缩函数相关

$$f_t(B, C, D) = \begin{cases} (B \wedge C) \vee ((\neg B) \wedge D) & \text{对于 } 0 \leq t \leq 19 \\ B \oplus C \oplus D & \text{对于 } 20 \leq t \leq 39 \\ (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) & \text{对于 } 40 \leq t \leq 59 \\ B \oplus C \oplus D & \text{对于 } 60 \leq t \leq 79 \end{cases}$$

Word Based Operation

■ 轮密钥 K_0, K_1, \dots, K_{79}

$$K_t = \begin{cases} 5A827999 & \text{对于 } 0 \leq t \leq 19 \\ 6ED9EBA1 & \text{对于 } 20 \leq t \leq 39 \\ 8F1BBCDC & \text{对于 } 40 \leq t \leq 59 \\ CA62C1D6 & \text{对于 } 60 \leq t \leq 79 \end{cases}$$

安全Hash算法

Compress 

密码体制 4.1 SHA-1(x)

external SHA-1-PAD

global K_0, \dots, K_{79}

$y \leftarrow \text{SHA-1-PAD}(x)$

令 $y = M_1 \parallel M_2 \parallel \dots \parallel M_n$, 其中每个 M_i 是一个 512 比特的分组

$H_0 \leftarrow 67452301$

$H_1 \leftarrow \text{EFCDAB89}$

$H_2 \leftarrow 98BADCFE$

$H_3 \leftarrow 10325476$

$H_4 \leftarrow \text{C3D2E1F0}$

for $i \leftarrow 1$ to n

 令 $M_i = W_0 \parallel W_1 \parallel \dots \parallel W_{15}$, 其中每个 W_t 是一个字

 for $t \leftarrow 16$ to 79

 do $W_t \leftarrow \text{ROTL}^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$

$A \leftarrow H_0$

$B \leftarrow H_1$

$C \leftarrow H_2$

$D \leftarrow H_3$

$E \leftarrow H_4$

 for $t \leftarrow 0$ to 79

 do {
 temp $\leftarrow \text{ROTL}^5(A) + f_t(B, C, D) + E + W_t + K_t$
 $E \leftarrow D$
 do {
 $D \leftarrow C$
 $C \leftarrow \text{ROTL}^{30}(B)$
 $B \leftarrow A$
 $A \leftarrow \text{temp}$

$H_0 \leftarrow H_0 + A$

$H_1 \leftarrow H_1 + B$

$H_2 \leftarrow H_2 + C$

$H_3 \leftarrow H_3 + D$

$H_4 \leftarrow H_4 + E$

return($H_0 \parallel H_1 \parallel H_2 \parallel H_3 \parallel H_4$)



安全Hash函数SHA-1

- 安全散列算法由NIST开发，1993年公布为联邦信息处理标准(FIPS 180)
- 最初载明的算法于 1993年发布，称做安全杂凑标准 (**Secure Hash Standard**), FIPS PUB 180。这个版本现在常被称为 SHA-0。它在发布之后很快就被 NSA 撤回，并且由 1995年发布的修订版本 FIPS PUB 180-1 (通常称为 SHA-1) 取代。SHA-1 和 SHA-0 的算法只在压缩函数的讯息转换部份差了一个位元的循环位移。
- Sha-1 产生160比特的散列值



安全Hash函数

- 2002 NIST制定了新版本标准 **FIPS-2**, 定义了三种新版本的**SHA**, 输出散列值分别为**256,384,512**比特(**SHA-2**)
- 新版本使用了与**SHA-1**相同的底层结构和相同类型的模运算以及相同的二元逻辑运算



安全Hash函数

- SHA-512(SHA-2)
- 以最大长度不超过 2^{128} 比特的消息为输入，生成**512**比特的消息摘要输出
- 输入以**1024**比特的数据块进行处理

安全Hash函数SHA-2

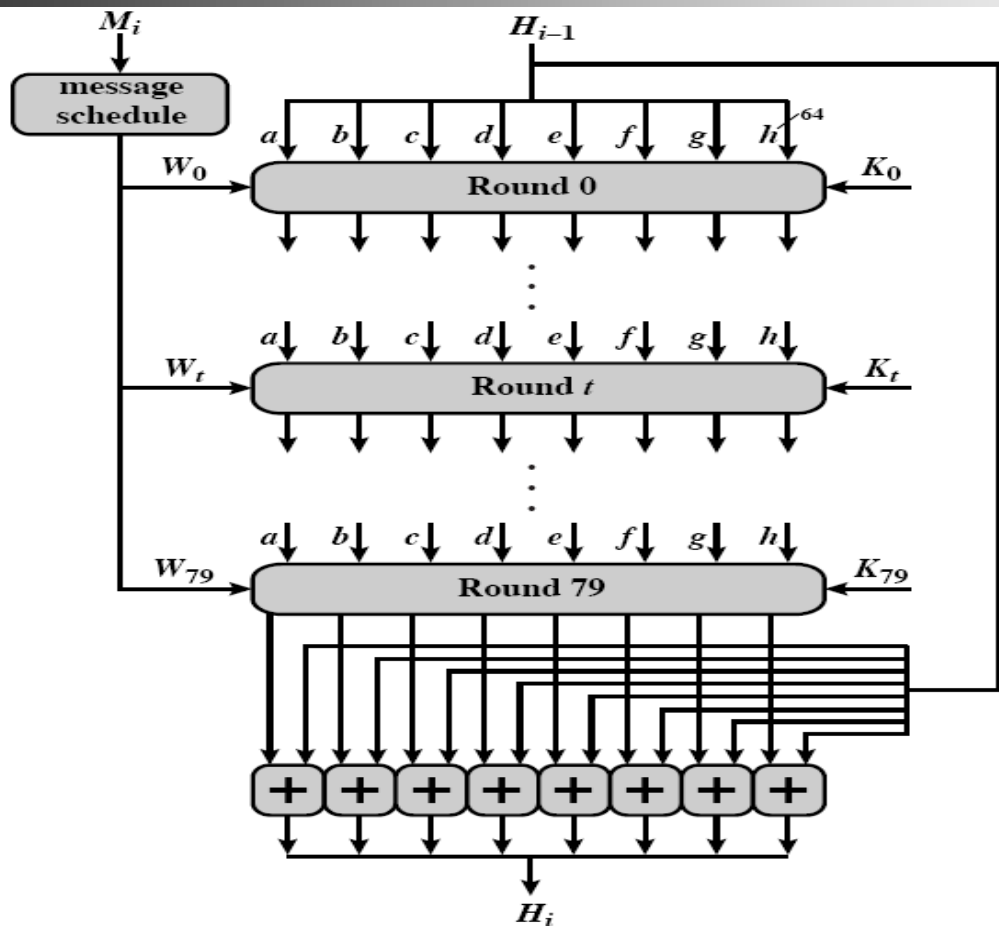


Figure 3.5 SHA-512 Processing of a Single 1024-Bit Block

安全Hash函数SHA-2

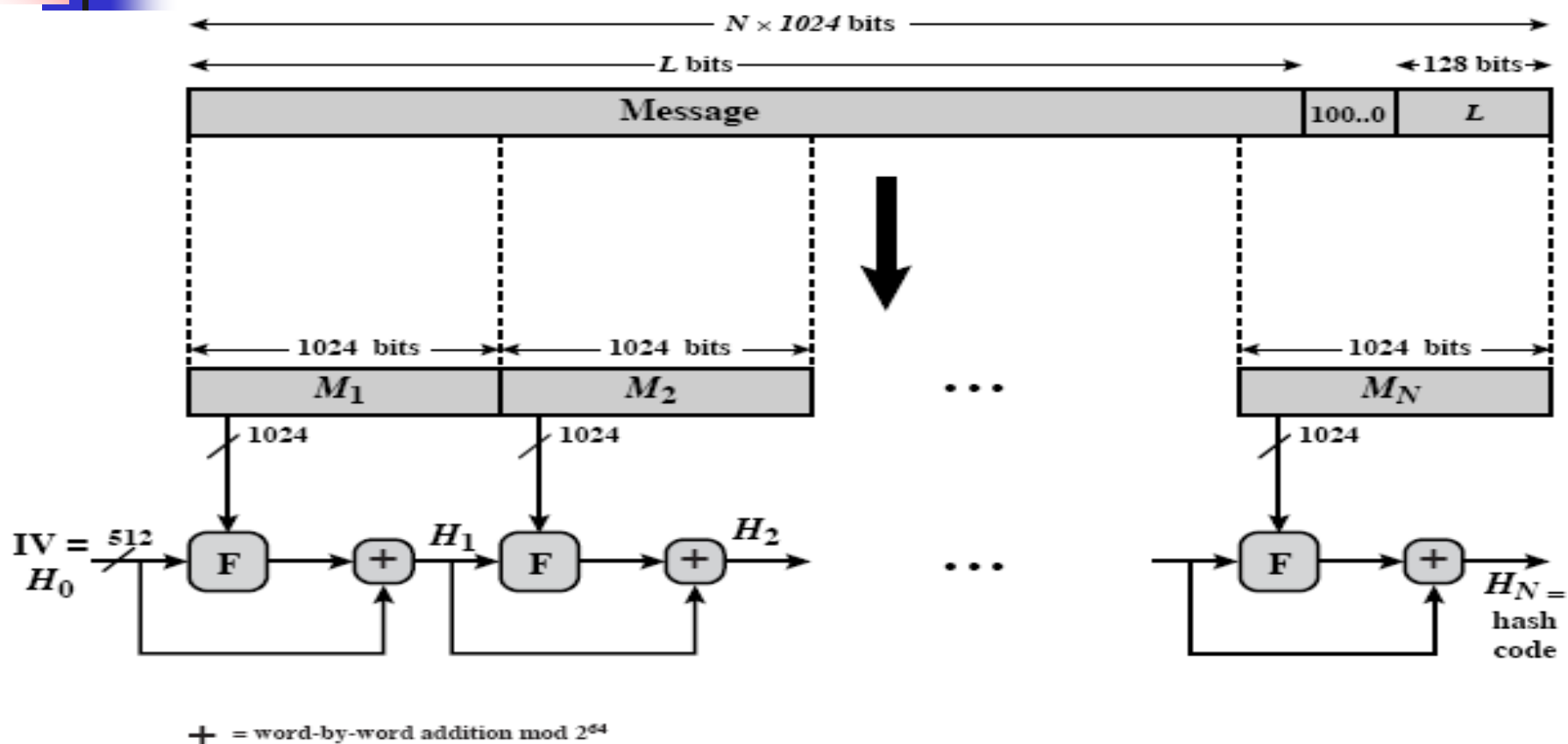


Figure 3.4 Message Digest Generation Using SHA-512



安全Hash函数

	SHA-1	SHA-256	SHA-384	SHA-512
Message digest size	160	256	384	512
Message size	$<2^{64}$	$<2^{64}$	$<2^{128}$	$<2^{128}$
Block size	512	512	1024	1024
Word size	32	32	64	64
Number of steps	80	64	80	80
Security	80	128	192	256

Sha参数的比较



安全Hash函数

- 类似的MD5等消息摘要算法都找到了有效的算法计算碰撞；
- 隐含着这一代的消息摘要算法都有内在的致命结构缺陷，直接加速了SHA-1的死亡，SHA-2的退出舞台；
- 2005， NIST宣布到2010不再认可SHA-1, 转为信任其他的消息摘要版本；
- SHA-3



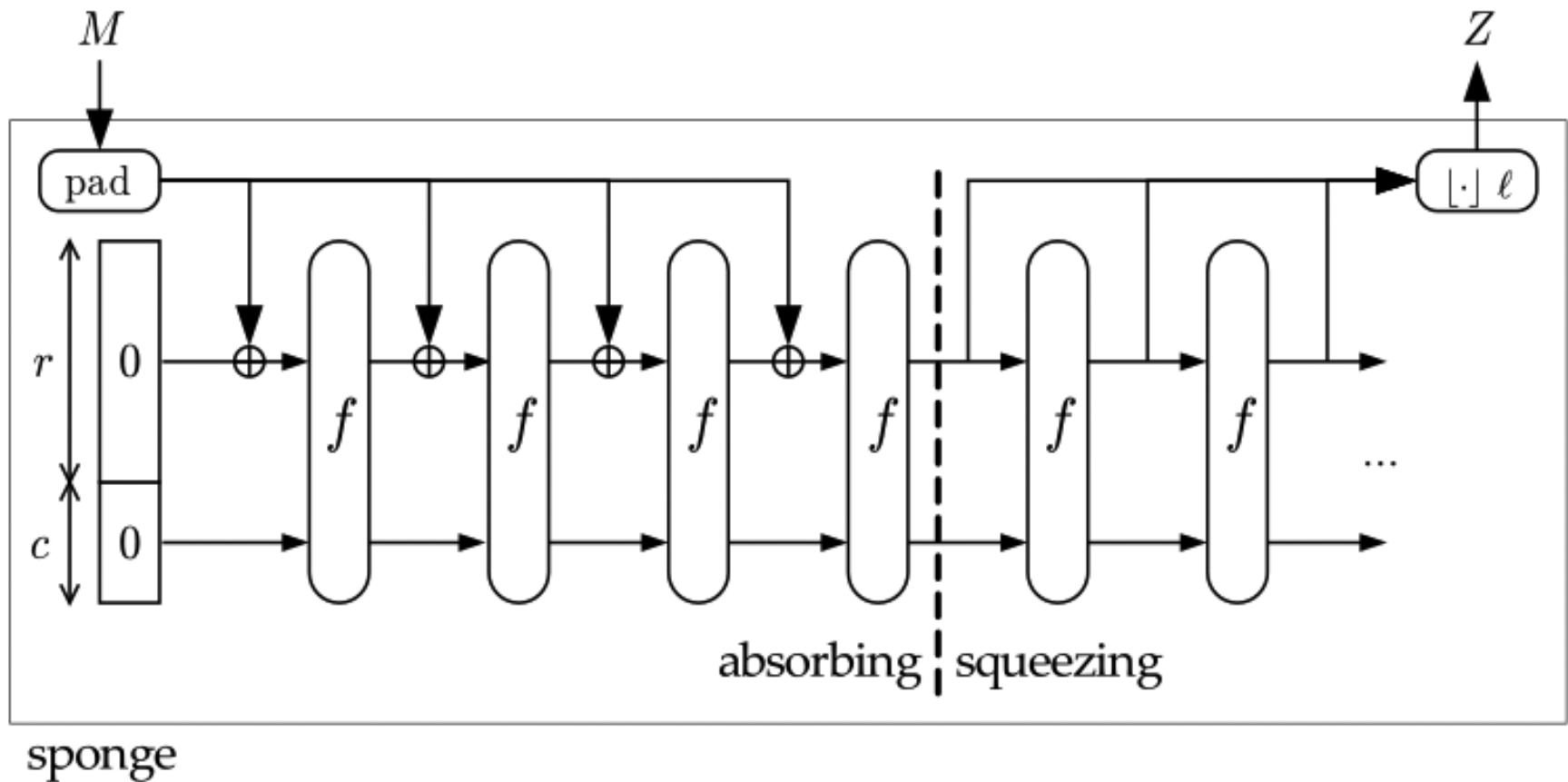
SHA-3: Third Round Candidates

- BLAKE
- Grøstl
- JH
- Keccak
- Skein

Function	256-bit output			512-bit output		
	Mbytes/s	cycles/byte	rel. SHA-2	Mbytes/s	cycles/byte	rel. SHA-2
SHA-2	138.37	17.30	1.000	178.36	13.42	1.000
BLAKE	173.74	13.78	1.256	295.80	8.09	1.658
Grøstl	82.09	29.16	0.593	51.23	46.73	0.287
JH	41.55	57.62	0.300	41.43	57.78	0.232
Keccak	140.91	16.99	1.018	75.62	31.66	0.424
Skein	340.87	7.02	2.463	344.15	6.96	1.930

SHA3 Structure

Keccak (SHA3-224, SHA3-256, SHA3-384, and SHA3-512)

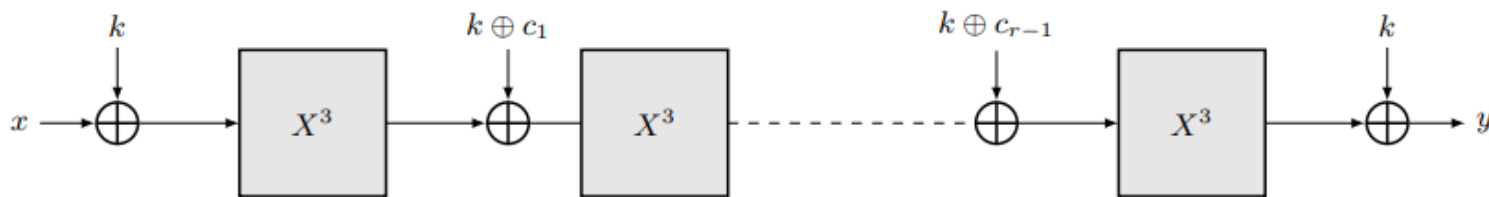


Zk-friendly Hash

- 区块链的应用反过来推进了Hash函数的研究
- 隐私保护—MIMC Hash, Poseidon Hash, ...

The **MiMC** hash function is specifically designed to minimize circuit size and thus ZKP cost by using only additions and multiplications.

Its simple *algebraic structure* is advantageous for applications which benefit from a low multiplicative complexity.



r rounds of MiMC- n/n



4.4 消息认证码

- MAC (Message Authentication Code)

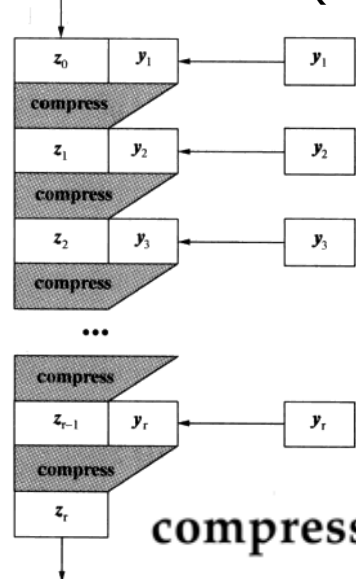
Keyed Hash

4.4 消息认证码

Direct Approach (安全隐患)

- 假定我们通过迭代Hash函数来构造一个新的带密钥的Hash函数 h_K

IV $IV=K$ (m bits)



假设输入消息 x 的长度是 t 的倍数; $|x'|=t$

那么 $h_K(x \parallel x') = \text{compress}(h_K(x) \parallel x')$

Length extension attack: *even K is secret*

$\text{compress} : \{0,1\}^{m+t} \rightarrow \{0,1\}^m$



4.4 消息认证码

■ HMAC

$$\text{HMAC}_K(x) = \text{SHA-1}((K \oplus \text{opad}) \parallel \text{SHA-1}((K \oplus \text{ipad}) \parallel x))$$

$$\text{ipad} = 3636 \dots 36$$

$$\text{opad} = 5C5C \dots 5C$$

■ CBC-MAC

Cryptosystem $\text{CBC-MAC}(x, K)$

denote $x = x_1 \parallel \dots \parallel x_n$

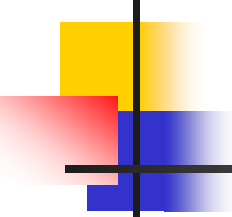
$\text{IV} \leftarrow 00 \dots 0$

$y_0 \leftarrow \text{IV}$

for $i \leftarrow 1$ **to** n

do $y_i \leftarrow e_K(y_{i-1} \oplus x_i)$

return (y_n)



4.5 无条件安全消息认证码

攻击者向Random Oracle提出 Q 个消息请求，获得一系列 $(x_1, y_1), (x_2, y_2), \dots, (x_Q, y_Q)$ 。
如果攻击者输出一个假冒的概率至少为 ϵ ，则攻击者对给定的MAC，称为一个 (ϵ, Q) 假冒者。

- 无条件安全：即使攻击者拥有无限的计算能力，不存在 $(\epsilon, 1)$ 假冒者, $(\epsilon, 0)$ 假冒者
- $(\epsilon, 0)$ --模仿(Impersonation)
- $(\epsilon, 1)$ --代替(Substitution)



4.5 无条件安全消息认证码

■ 例子

例 假定

$$\mathcal{X} = \mathcal{Y} = \mathbb{Z}_3$$

且

$$\mathcal{K} = \mathbb{Z}_3 \times \mathbb{Z}_3$$

对每一个 $K = (a, b) \in \mathcal{K}$ 和每一个 $x \in \mathcal{X}$ ，定义

$$h_{(a,b)}(x) = ax + b \bmod 3$$

再定义

$$\mathcal{H} = \{h_{(a,b)} : (a,b) \in \mathbb{Z}_3 \times \mathbb{Z}_3\}$$



4.5 无条件安全消息认证码

key	0	1	2
(0, 0)	0	0	0
(0, 1)	1	1	1
(0, 2)	2	2	2
(1, 0)	0	1	2
(1, 1)	1	2	0
(1, 2)	2	0	1
(2, 0)	0	2	1
(2, 1)	1	0	2
(2, 2)	2	1	0

认证矩阵

4.5 无条件安全消息认证码

key	0	1	2
(0, 0)	0	0	0
(0, 1)	1	1	1
(0, 2)	2	2	2
(1, 0)	0	1	2
(1, 1)	1	2	0
(1, 2)	2	0	1
(2, 0)	0	2	1
(2, 1)	1	0	2
(2, 2)	2	1	0

Pd_Q : the probability ϵ that an adversary can create a successful forgery after observing Q valid message-tag pairs.

$$Pd_0 = 1/3$$

$$Pd_1 = 1/3$$



强泛Hash函数族

■ Strongly Universal Hash Families

Definition Suppose that $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$ is an (N, M) hash family. This hash family is *strongly universal* provided that the following condition is satisfied for every $x, x' \in \mathcal{X}$ such that $x \neq x'$, and for every $y, y' \in \mathcal{Y}$:

$$|\{K \in \mathcal{K} : h_K(x) = y, h_K(x') = y'\}| = \frac{|\mathcal{K}|}{M^2}.$$