



# 密码学

## 第六章 哈希函数

网络空间安全学院

朱丹 戚明平

zhudan/mpqi@nwpu.edu.cn

- ✦ 一些国际标准化组织已把椭圆曲线密码作为新的信息安全标准。如，IEEE P1363/D4，ANSI F9.62，ANSI F9.63等标准，分别规范了椭圆曲线密码在Internet协议安全、电子商务、Web服务器、空间通信、移动通信、智能卡等方面的应用
- ✦ 它密钥短，软件实现规模小、硬件实现节省电路。由于椭圆曲线离散对数问题尚没有发现亚指数算法，所以普遍认为，椭圆曲线密码比RSA和ElGamal密码更安全
- ✦ 椭圆曲线密码已成为RSA之外呼声最高的公钥密码之一

- ✦ 160位的椭圆曲线密码的安全性相当于1024位的RSA密码，而且运算速度也较快
- ✦ ElGamal密码建立在有限域 $GF(p)$ 的乘法群的离散对数问题的困难性之上。而椭圆曲线密码建立在椭圆曲线群的离散对数问题的困难性之上。两者的主要区别是其离散对数问题所依赖的群不同。因此两者有许多相似之处
- ✦ 基于 $GF(p)$ 和 $GF(2^m)$ 上的椭圆曲线，都可以构成安全的椭圆曲线密码
- ✦ 我国商用密码采用了椭圆曲线密码，并具体颁布了椭圆曲线密码标准算法SM2

## ✎ $GF(p)$ 上椭圆曲线密码算法

### ✎ $GF(p)$ 上椭圆曲线密码的基础参数

- $T = \langle p, a, b, G, n, h \rangle$ ,  $p$ 为大于3素数,  $p$ 确定了有限域 $GF(p)$
- 元素 $a, b \in GF(p)$ ,  $a$ 和 $b$ 确定了椭圆曲线:  $y^2 = x^3 + ax + b$ ,  $a, b \in GF(p)$
- $G$ 为循环子群 $E_1$ 的生成元,  $n$ 为素数且为生成元 $G$ 的阶,  $G$ 和 $n$ 确定了循环子群 $E_1$
- $h = \frac{|E|}{n}$ , 并称为余因子,  $h$ 将交换群 $E$ 和循环子群 $E_1$ 联系起来

## ✎ $GF(p)$ 上椭圆曲线密码算法

### ✎ $GF(p)$ 上椭圆曲线密码的密钥

- 用户的私钥定义为一个随机数 $d$ ,  $d \in \{1, 2, \dots, n-1\}$
- 用户的公钥定义为 $Q$ 点,  $Q = dG$
- 由公钥 $Q$ 求私钥 $d$ 是求解椭圆曲线离散对数问题, 当 $p$ 足够大时, 这是困难的

## ✎ $GF(p)$ 上椭圆曲线密码算法

✎ 设 $d$ 为用户私钥,  $Q$ 为用户公钥, 明文数据为 $M$ ,  $0 \leq M \leq n - 1$

### ✎ 加密过程

- S1: 选择一个随机数 $k$ , 且 $k \in \{1, 2, \dots, n - 1\}$
- S2: 计算点 $X_1(x_1, y_1) = kG$
- S3: 计算点 $X_2(x_2, y_2) = kQ$ , 如果分量 $x_2 = 0$ , 则转S1
- S4: 计算密文 $C = Mx_2 \bmod n$
- $(X_1, C)$ 为最终的密文数据

## ✎ $GF(p)$ 上椭圆曲线密码算法

✎ 设 $d$ 为用户私钥,  $Q$ 为用户公钥, 密文数据为 $(X_1, C)$

## ✎ 解密过程

➤ S1: 利用私钥 $d$ 求出 $X_2(x_2, y_2)$

$$dX_1 = d(kG) = k(dG) = kQ = X_2$$

➤ S2: 利用 $X_2(x_2, y_2)$ 计算得到明文 $M$

$$M = Cx_2^{-1} \bmod n$$

✎ 推荐使用256位素域 $GF(p)$ 上的椭圆曲线  $y^2 = x^3 + ax + b$ , 曲线参数如下:

$p = \text{FFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF}$   
 $a = \text{FFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFFC}$   
 $b = \text{28E9FA9E 9D9F5E34 4D5A9E4B CF6509A7 F39789F5 15AB8F92 DDBCBD41 4D940E93}$   
 $n = \text{FFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF 7203DF6B 21C6052B 53BBF409 39D54123}$   
 $x_G = \text{32C4AE2C 1F198119 5F990446 6A39C994 8FE30BBF F2660BE1 715A4589 334C74C7}$   
 $y_G = \text{BC3736A2 F4F6779C 59BDCEE3 6B692153 D0A9877C C62A4740 02DF32E5 2139F0A0}$

✎ 密钥:

✎ 私钥是随机数 $d$ ,  $d \in [1, n - 1]$

✎ 公钥 $P = dG = d(x_G, y_G)$



## ✎ SM2的加密算法:

- ✎ S1: 产生随机数 $k$ ,  $1 \leq k \leq n - 1$
- ✎ S2: 计算椭圆曲线点 $C_1 = kG = (x_1, y_1)$
- ✎ S3: 计算椭圆曲线点 $kP = (x_2, y_2)$
- ✎ S4: 计算 $t = KDF(x_2 \| y_2, klen)$ , 若 $t$ 为全0比特串, 则返回S1  
// $KDF(Z, klen)$ 是密钥派生函数, 它利用HASH函数从数据 $Z$ 产生出长度为 $klen$ 的密钥数据
- ✎ S5: 计算 $C_2 = M \oplus t$ ;
- ✎ S6: 计算 $C_3 = Hash(x_2 \| M \| y_2)$
- ✎ S7: 输出密文 $C = C_1 \| C_2 \| C_3$

## ✎ SM2的解密算法:

- ✎ S1: 计算 $dC_1 = (x_2, y_2)$
- ✎ S2: 计算椭圆曲线点 $t = KDF(x_2 \| y_2, klen)$ , 若 $t$ 为全0比特串, 则报错退出
- ✎ S3: 计算椭圆曲线点 $M' = C_2 \oplus t$
- ✎ S4: 输出明文 $M'$
- ✎ S5: 计算 $C'_3 = Hash(x_2 \| M' \| y_2)$ , 判断 $C'_3$ 与 $C_3$ 是否相等 (验证)

## ✎ SM2的算法应用:

- ✎ 我国二代居民身份证采用了SM2椭圆曲线密码
- ✎ 我国的许多商用系统采用了SM2椭圆曲线密码

## ✦ 传统ECC与SM2的比较:

### 传统ECC

- ✦ 计算点 $X_2 (x_2, y_2) = kQ$
- ✦ 计算密文 $C = Mx_2 \bmod n$
- ✦ 最终密文数据 $(X_1, C)$

### SM2

- ✦ 计算点 $t = KDF(x_2 \| y_2, klen)$
- ✦ 计算密文 $C_2 = M \oplus t$
- ✦ 最终密文数据 $C = C_1 \| C_2 \| C_3$

- ✦ 传统ECC用 $x_2$ 加密, 加密是乘法; SM2处理 $(x_2, y_2)$ 产生 $t$ , 用 $t$ 加密, 加密是模2加
- ✦  $t$ 与 $(x_2, y_2)$ 相关, 更安全。SM2利用 $C_3$ 的验证, 进一步确保密文和解密的正确性
- ✦ SM2的密文数据扩展较大

## ✎ Hash函数的概念

- ✎ Hash函数又称哈希函数、杂凑函数、散列函数等，它能够将“任意长度”的消息变为某一固定长度的消息摘要（也称数字指纹、杂凑值、散列值等）。通常记为  $h = H(M)$  或  $Hash(M)$ 。

## ✎ Hash函数的性质

- ✎ Hash函数的输入可以是“任意长度”的消息；
- ✎ Hash函数的输出位长固定，多数情况下输入的长度是大于输出的长度的，因此Hash函数具有压缩特性；
- ✎ 有效性，即对给定的 $m$ ，计算 $h = H(m)$ 的运算是高效的；
- ✎ 具有极强的错误检测能力，即输入有很小的不同，输出将有很大的不同

## ✦ Hash函数的定义

- ✦ Hash函数将任意长的数据 $M$ 变换为定长的码 $h$ ，记为：

$$h = H(M) \text{ 或者 } h = Hash(M)$$

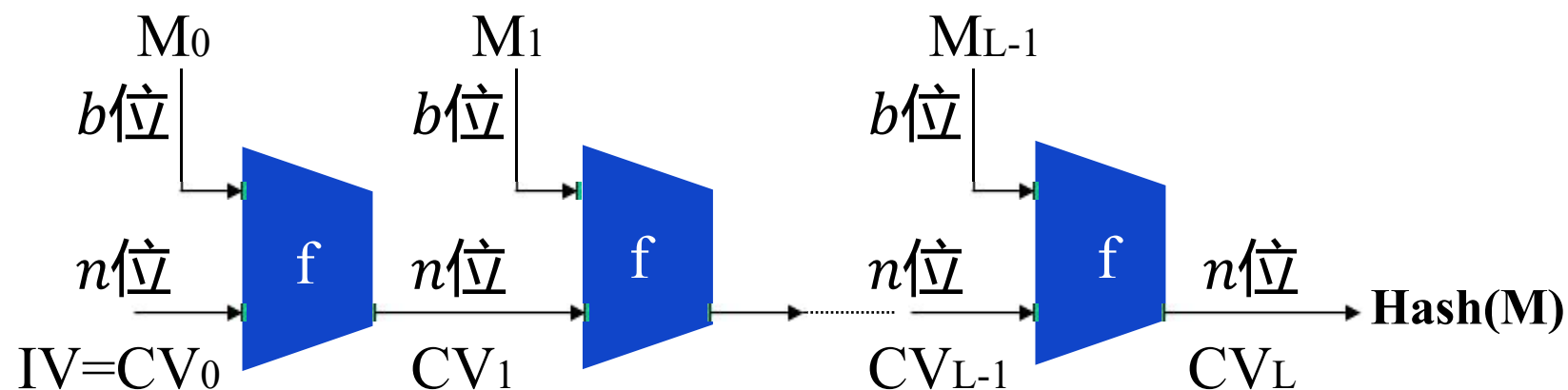
一般， $h$ 的长度小于 $M$ 的长度，因此**HASH函数是一种压缩变换**

- ✦ 安全性：

- **随机性**：哈希函数的输出具有**伪随机性**
- **单向性**：对给定的Hash值 $h$ ，找到满足 $H(x) = h$ 的 $x$ **在计算上是不可行的**
- **抗弱碰撞性**：对任何给定的 $x$ ，找到满足 $y \neq x$ 且 $H(x) = H(y)$ 的 $y$ **在计算上是不可行的**
- **抗强碰撞性**：找到任何满足 $H(x) = H(y)$ 的偶对 $(x, y)$ **在计算上是不可行的**

## 安全Hash函数处理数据的一般模型

Merkle最早提出了Hash函数处理数据 $M$ 的一般模型，其中 $M = M_0 \parallel \dots \parallel M_{L-1}$



$b$ 为分组长度， $f$ 为压缩函数， $L$ 为链接迭代轮数， $n$ 为输出位数。

## ✎ 安全Hash函数处理数据的一般模型

- ✎ 分组：将输入 $M$ 分为 $L - 1$ 个大小为 $b$ 位的分组
- ✎ 填充：若第 $L - 1$ 个分组不足 $b$ 位，则将其填充为 $b$ 位
- ✎ 附加：再附加上一个输入的总长度

填充和附加之后，共 $L$ 个大小为 $b$ 位的分组。

由于输入中包含长度，所以攻击者必须找出具有相同Hash值且长度相等的两条报文，或者找出两条长度不等但加入报文长度后Hash值相同的报文，从而增加了攻击的难度。

目前大多数Hash函数均采用这种数据处理模型。

# 章节安排

Outline



哈希函数简介

---



SHA-1哈希函数

---



SM3哈希函数

---



基于哈希的消息认证码HMAC

---



# 章节安排

Outline



哈希函数简介

---



SHA-1哈希函数

---



SM3哈希函数

---



基于哈希的消息认证码HMAC

---

### ✎ 安全哈希函数的应用-认证

#### ✎ 报文认证 (方案a)

- $A \rightarrow B: \langle M \parallel E(H(M), K) \rangle$
- 发送方A生成消息 $M$ 的Hash码 $H(M)$ 并使用传统密码对其加密, 将加密后的结果附于消息 $M$ 之后发送给接收方
- 由于 $H(M)$ 受密码保护, 所以接收方B可以通过比较 $H(M)$ 可认证报文的真实性和完整性

### ✎ 安全哈希函数的应用-认证

#### ✎ 报文认证 (方案b)

- $A \rightarrow B: \langle M \parallel H(M \parallel S) \rangle$
- 发送方A生成消息 $M$ 和秘密值 $S$ 的Hash码 $H(M)$ ，并将结果附于消息 $M$ 之后发送给接收方B
- 假定通信双方AB共享公共的秘密值 $S$ ，B可以通过验证Hash码来认证数据的真实性和完整性
- 该方案无需加密，因为秘密值 $S$ 参与Hash的计算



方案b有什么优点

### ✎ 安全哈希函数的应用-认证和保密

#### ✎ 报文认证和保密（方案a）

- $A \rightarrow B: \langle E(M \parallel H(M), K) \rangle$
- 发送方A生成消息M的Hash码 $H(M)$ ，并将结果附于消息M之后利用密钥加密并将密文发送给接收方B
- 由于只有A和B共享秘密密钥，所以B通过比较 $H(M)$ 可认证报文源和报文的真实性
- 由于该方法是对整个报文M和Hash码加密，所以也提供了保密性

### ✎ 安全哈希函数的应用-认证和保密

#### ✎ 报文认证和保密（方案b）

- $A \rightarrow B: \langle E(M \parallel H(M \parallel S), K) \rangle$
- 发送方A生成消息 $M$ 和秘密值 $S$ 的Hash码 $H(M)$ ，并将结果附于消息 $M$ 之后利用密钥加密并将密文发送给接收方B
- 由于只有A和B共享秘密密钥 $K$ 以及秘密值 $S$ ，所以B通过比较 $H(M \parallel S)$ 可认证报文源和报文的真实性
- 由于该方法是对整个报文 $M$ 和Hash码加密，所以也提供了保密性

### ✎ 安全哈希函数的应用-认证和数字签名

#### ✎ 报文认证和数字签名

- $A \rightarrow B: \langle M \parallel D(H(M), K_{dA}) \rangle$
- 发送方A使用公钥密钥用其私钥 $K_{dA}$ 对消息 $M$ 的Hash码 $H(M)$ 签名, 并将结果附于消息 $M$ 之后发送给接收方B
- B可以验证Hash值来认证报文的真实性, 因为该方法可提供认证
- 由于只有A可以进行签名, 所以也提供了数字签名

### ✎ 安全哈希函数的应用-认证、数字签名和保密

#### ✎ 报文认证、数字签名和保密

- $A \rightarrow B: \langle E(M \parallel D(H(M), K_{dA}), K) \rangle$
- 发送方A使用公钥密钥用其私钥 $K_{dA}$ 对消息 $M$ 的Hash码 $H(M)$ 签名, 并将结果附于消息 $M$ 之后用传统密码对 $M$ 和签名进行加密, 发送给接收方B
- B可以验证Hash值来认证报文的真实性, 因为该方法可提供认证
- 由于只有A可以进行签名, 所以也提供了数字签名
- 由于该方法是对消息和Hash码的签名整体进行加密, 所以也提供了保密性

# 章节安排

Outline



哈希函数简介

---



SHA-1哈希函数

---



SM3哈希函数

---



基于哈希的消息认证码HMAC

---



- ✎ 安全哈希函数的设计同分组密码一样复杂困难，一些国际组织和密码学者不断制定和颁布Hash函数标准，包括
  - ✎ Rivest——MD4, MD5, MD6
  - ✎ NIST——SHA-0, SHA-1, SHA-2, SHA-3
  - ✎ 欧洲——RipeMD, Whirlpool
  - ✎ 澳大利亚——HAVAL
  - ✎ 中国——SM3

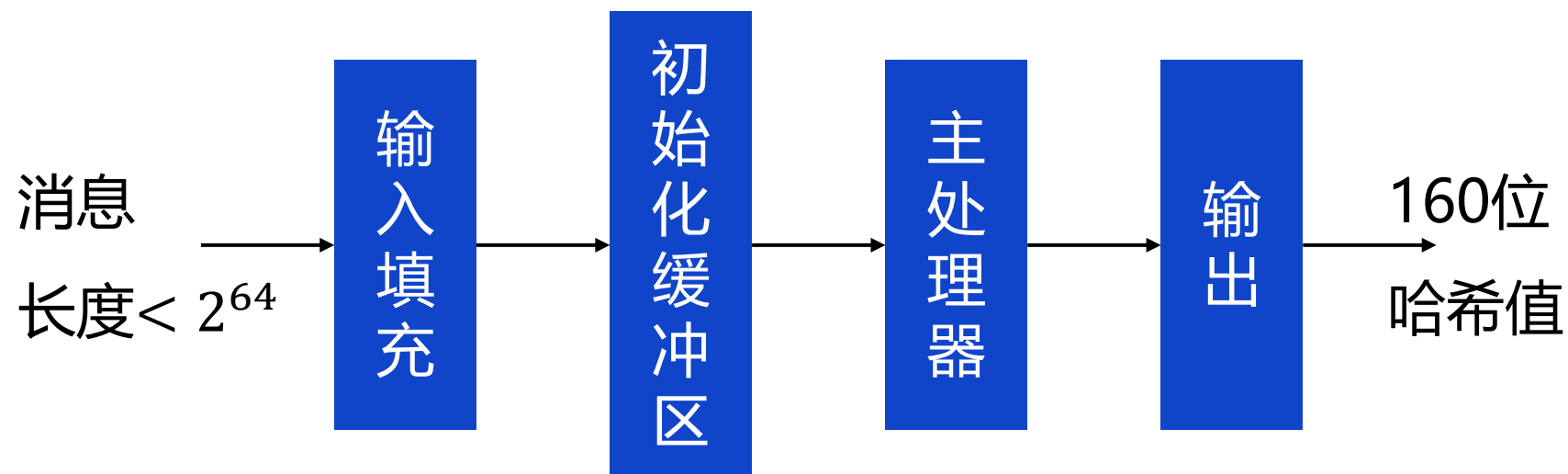
### ✎ SHA系列哈希函数：

- ✎ **SHA系列Hash函数**由美国标准与技术研究所(NIST)设计
- ✎ 1993年公布了**SHA-0** (FIPS PUB 180), 后来发现它不安全;
- ✎ 1995年又公布了**SHA-1** (FIPS PUB 180-1); 【2017年Google给出第一个碰撞】
- ✎ 2002年又公布了**SHA-2** (FIPS PUB 180-2), **SHA-2**包括3个Hash算法: **SHA-256, SHA-384, SHA-512**; 【2008年补充了**SHA-224**】
- ✎ 2005年, 王小云院士给出了一种攻击**SHA-1**的方法, 用 $2^{69}$ 次操作找到一个强碰撞, 以前认为是穷举 (生日) 攻击 $2^{80}$ 次操作
- ✎ NIST于2007年公开征集**SHA-3**, 并于2012公布SHA-3获胜算法为**Keccak**

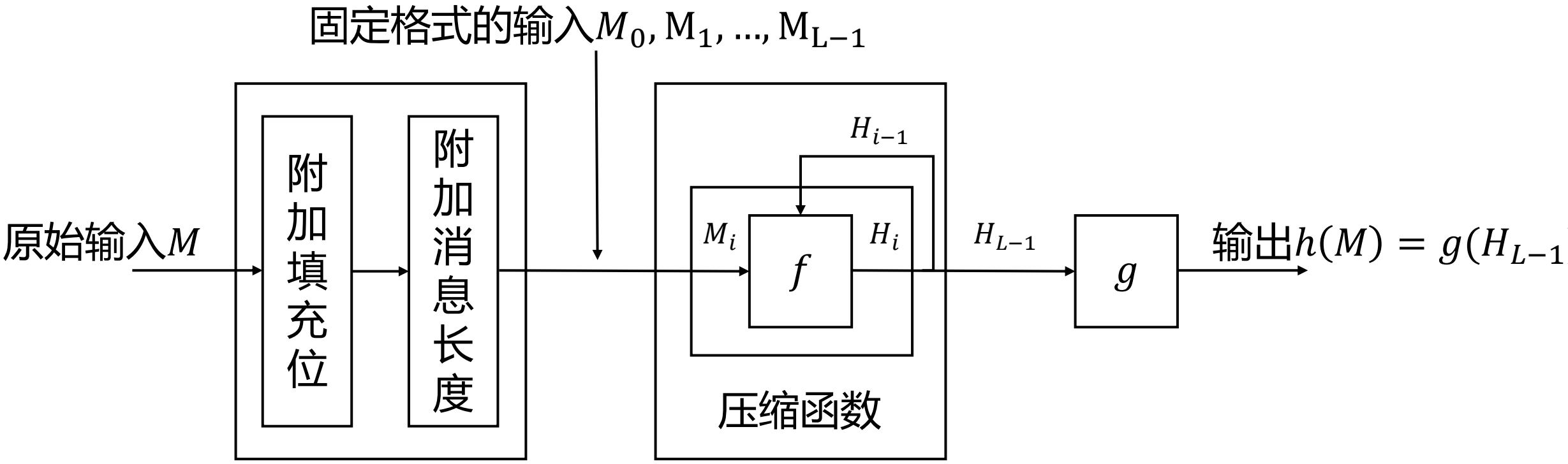
 各版本SHA算法参数比较:

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
消息摘要长度	160	224	256	384	512
消息长度	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
分组长度	512	512	512	1024	1024
字长度	32	32	32	64	64
步骤数	80	64	64	80	80
穷尽安全性	80	112	128	192	256

✎ SHA-1的结构：采用Merkle提出的安全Hash模型



✎ SHA-1算法



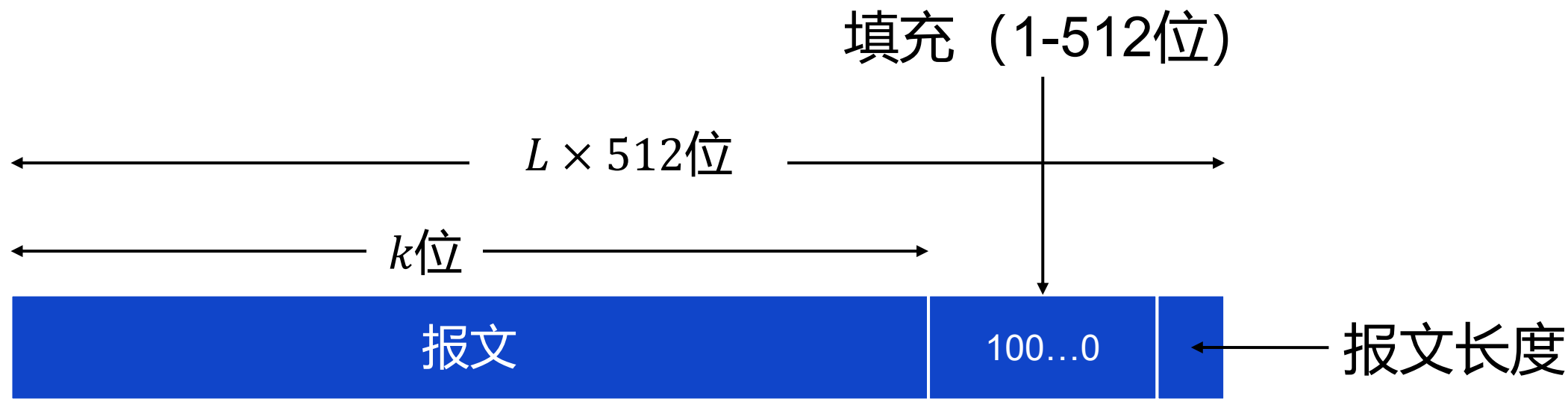
### ✎ SHA-1算法的输入填充

- ✎ 目的是使填充后的报文长度满足：

$$\text{长度} = 448 \bmod 512$$

- 填充方法是在报文后附加一个1和若干个0
- 然后附上表示填充前报文长度的64位数据（最高有效位在前）
- ✎ 若报文本身已经满足上述要求，仍然需要填充（例如，若报文长度为448位，则仍需填充512位使其长度为960位），因此填充位数在1到512位之间
- ✎ 经过填充和附加后，数据的长度为512位的整数倍

✎ SHA-1算法的输入填充



对消息进行填充，使之加上64位长度信息后，总长度为512的整数倍

### SHA-1算法的初始化缓冲区

- 缓冲区由5个32位的寄存器（A, B, C, D, E）组成，用于保存160位的中间结果和最终结果
- 将寄存器初始化为下列32位的整数：

A: 67452301

B: EFCDAB89

C: 98BADCFE

D: 10325476

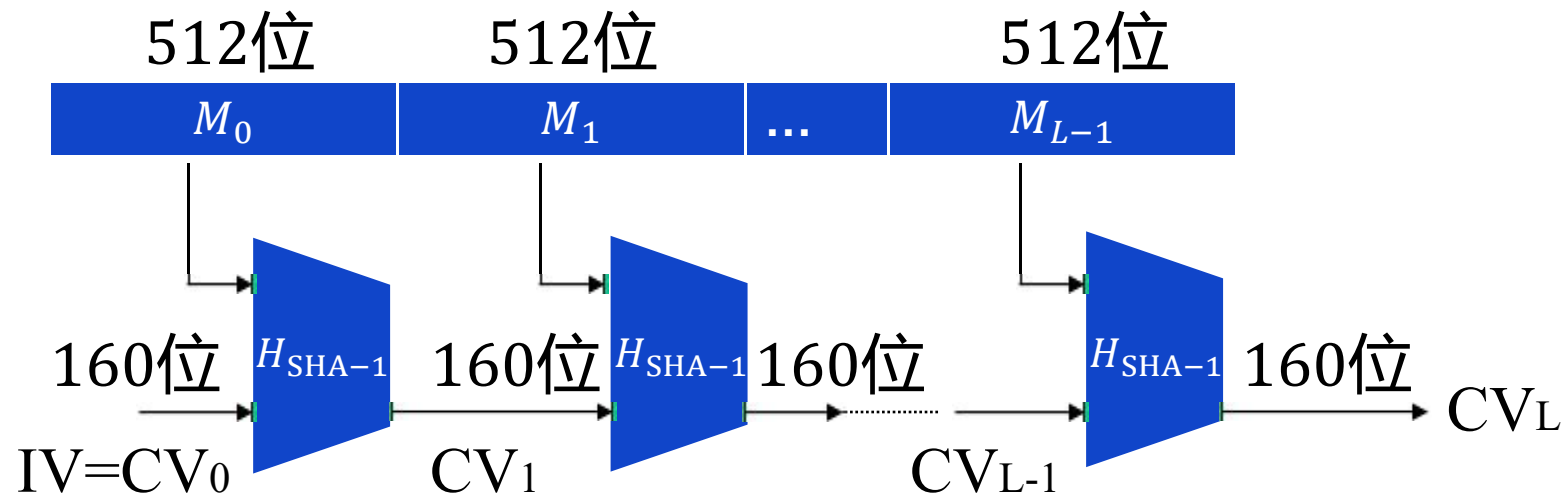
E: C3D2E1F0

注意：高有效位存于低地址



### ✎ SHA-1算法的主处理

- ✎ 主处理是SHA-1 HASH函数的核心
- ✎ 每次处理一个512位的分组，链接迭代处理（填充后报文）的所有 $L$ 个分组数



利用SHA-1算法产生报文摘要 $CV_L$ ，其中 $H_{SHA-1}$ 是SHA-1算法的压缩函数

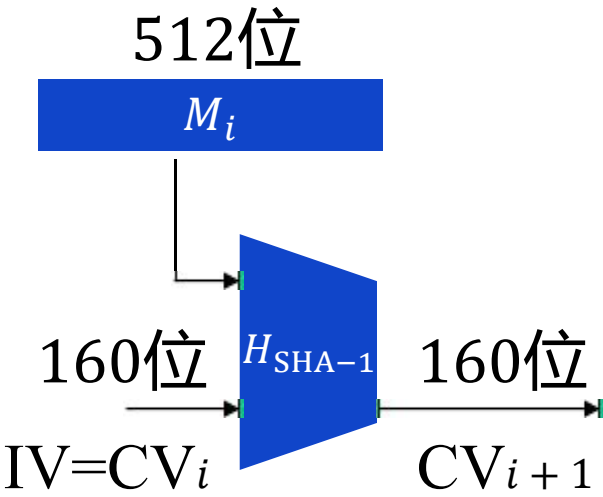
✎ SHA-1算法的主处理

✎ 将填充后的消息 $M$ 按512比特进行分组, 记为  $M = M_0 \parallel M_1 \parallel \dots \parallel M_{L-1}$  ,  
其中 $L = (K + n + 65)/512$ ,  $n$ 为填充时插入的0的个数

✎ 对分组进行迭代压缩处理

$$\begin{cases} CV_0 = IV, IV \text{为预设的初始值 (160 bit)} \\ CV_{i+1} = H_{SHA-1}(CV_i, M_i), 0 \leq i \leq L - 1 \\ H(M) = CV_L \end{cases}$$

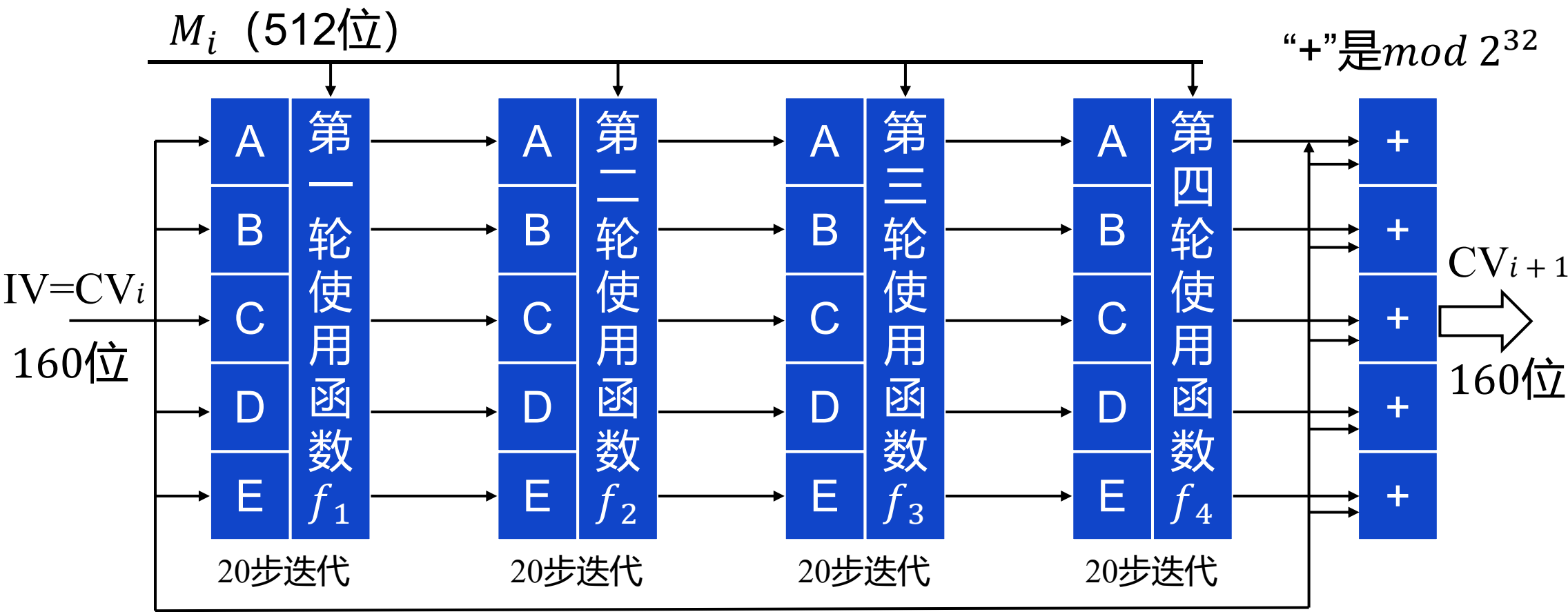
A=0x67452301  
B=0xEFCDAB89  
C=0x98BADCFE  
D=0x10325476  
E=0xC3D2E1F0



### ✎ SHA-1算法的主处理

- ✎ 压缩函数是主处理的核心
- ✎ 它由四层运算（每层迭代20步）组成，四层的运算结构相同
- ✎ 每轮的输入是当前要处理的512位的分组 $M_i$ 和160位缓冲区ABCDE的内容，每轮都对ABCDE的内容更新，而且每轮使用的逻辑函数不同，分别为 $f_1$ ， $f_2$ ， $f_3$ 和 $f_4$
- ✎ 第四轮的输出与第一轮的输入相加得到压缩函数的输出

✎ SHA-1算法的主处理



### SHA-1算法的主处理

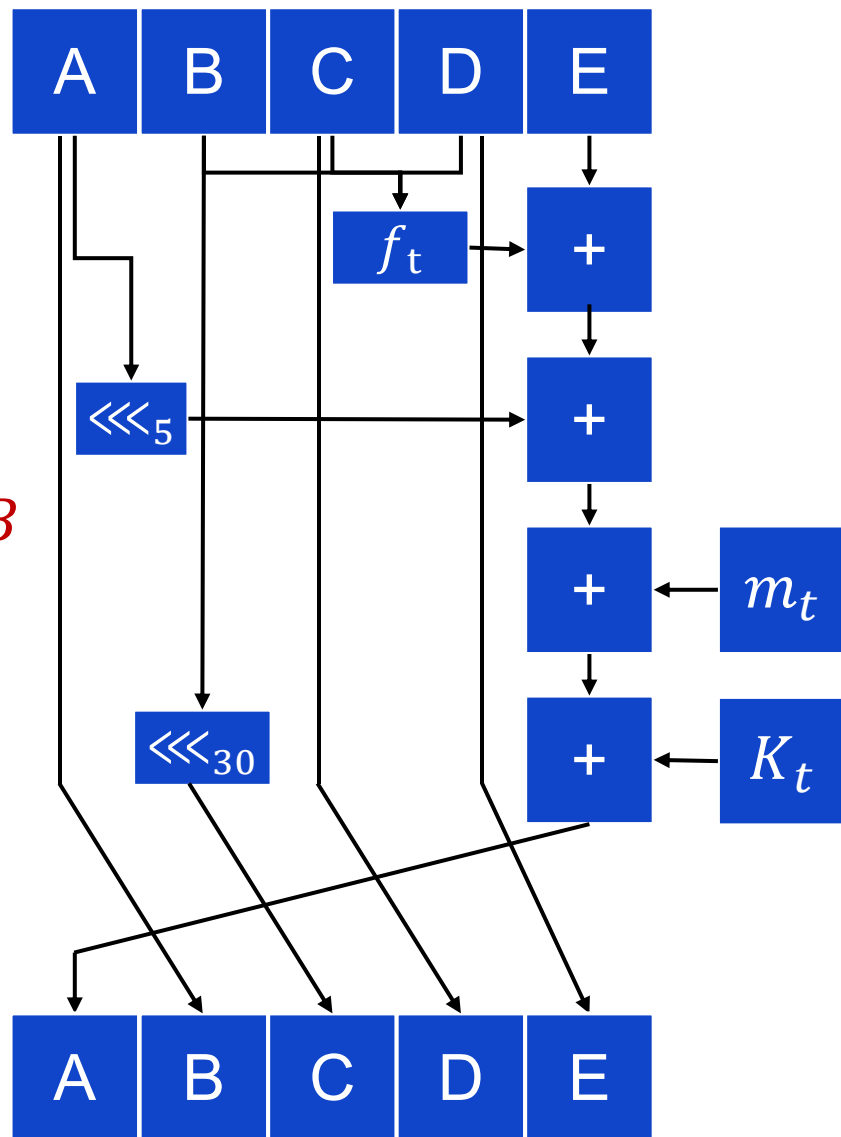
#### SHA-1算法压缩函数中的单步操作

➤  $(A, B, C, D, E)$

$$= ((E + f_t(B, C, D) + (A \lll 5) + m_t + K_t, A, B \lll 30, C, D)$$

➤ 逻辑函数:  $f_1 = (B \wedge C) \vee (\neg B \wedge D)$ ,  $f_2 = B \oplus C \oplus D$ ,

$$f_3 = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D), \quad f_4 = B \oplus C \oplus D$$



## SHA-1算法的主处理

## SHA-1算法压缩函数中的单步操作

▶ 常量:  $K_t = 0x5A827999 \quad 0 \leq t \leq 19$

$$K_t = 0x6ED9EBA1 \quad 20 \leq t \leq 39$$

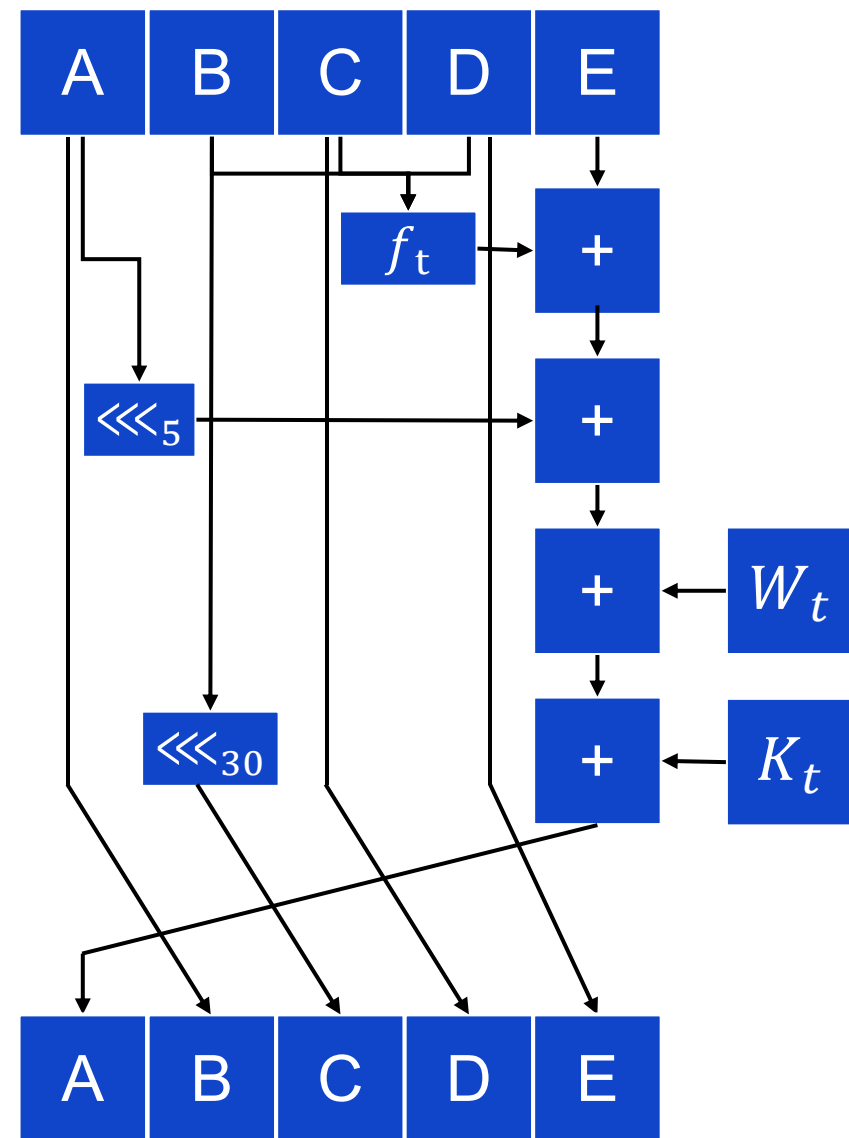
$$K_t = 0x8F1BBCDC \quad 40 \leq t \leq 59$$

$$K_t = 0xCA62C1D6 \quad 60 \leq t \leq 79$$

### 512位数据分组扩展:

当 $0 \leq t \leq 15$ 时,  $W_t = m_t$ ; 当 $16 \leq t \leq 79$ 时,

$$m_t = (m_{t-16} \oplus m_{t-14} \oplus m_{t-8} \oplus m_{t-3}) \lll 1$$



# 章节安排

Outline



哈希函数简介

---



SHA-1哈希函数

---



SM3哈希函数

---



基于哈希的消息认证码HMAC

---

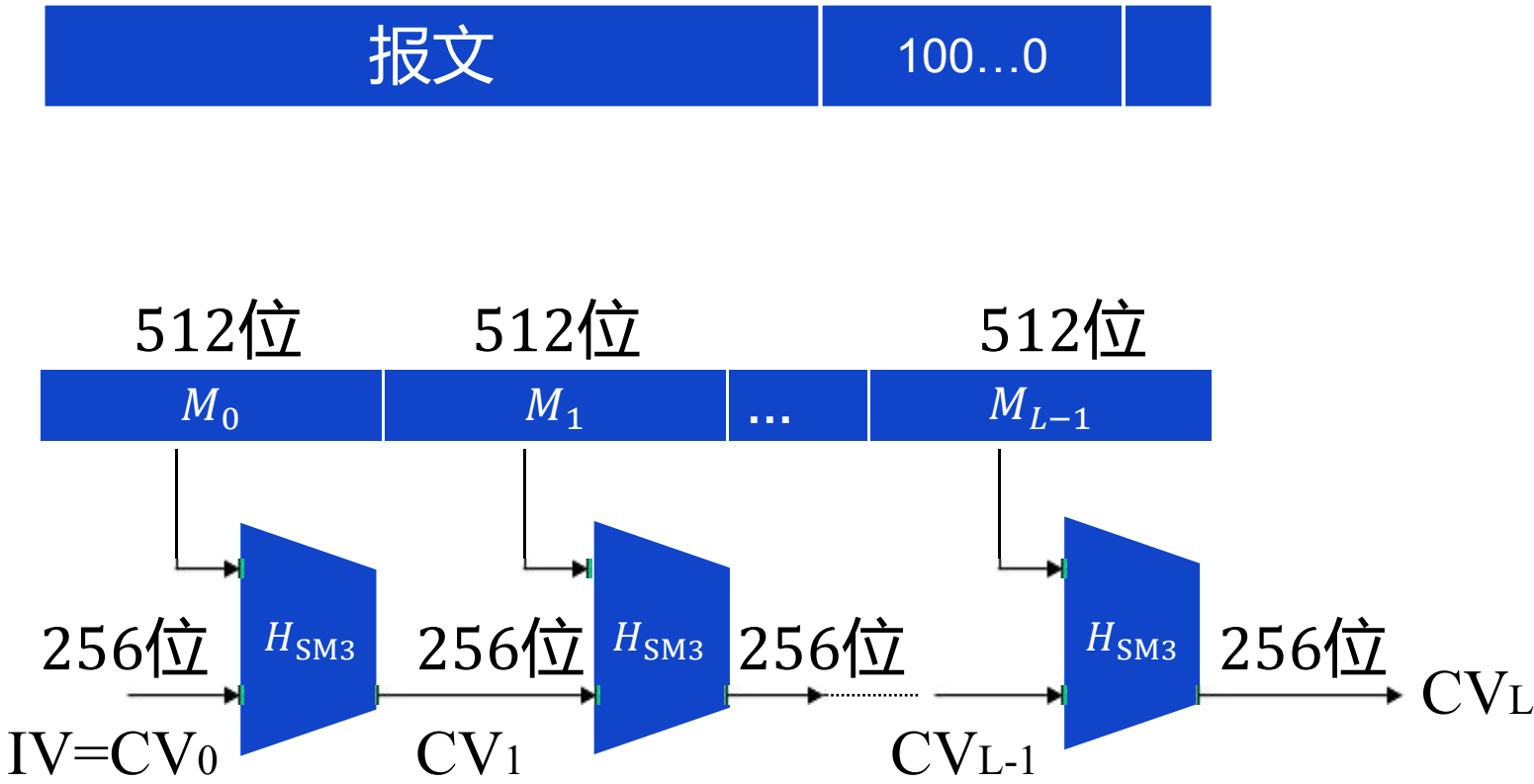
- ✎ SM3哈希函数中国国家密码管理局颁布的一种哈希函数
  - ✎ 2010年12月国家密码管理局正式颁布
  - ✎ 适用于商用密码应用中的数字签名和验证、消息验证码的生成与验证以及随机数的生成
  - ✎ 可满足多种密码应用的安全需求
  - ✎ 面向32bit的字设计，数据分组长度为512 bit，输出Hash值位长为256 bit
  - ✎ SM3标准文档

<http://www.oscca.gov.cn/sca/xxgk/2010-12/17/1002389/files/302a3ada057c4a73830536d03e683110.pdf>



✎ SM3 哈希函数的 “压缩函数” + “迭代结构”

✎ 与SHA-1 哈希函数相同



✎ SM3哈希函数的 “压缩函数” + “迭代结构”

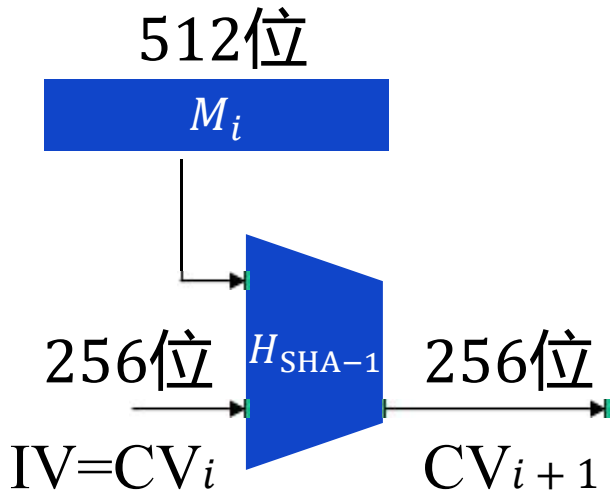
✎ 与SHA-1哈希函数相同

✎ 将填充后的消息 $M$ 按512比特进行分组，记为  $M = M_0 \parallel M_1 \parallel \dots \parallel M_{L-1}$ ，其中  $L = (K + n + 65)/512$ ， $n$ 为填充时插入的0的个数

✎ 对分组进行迭代压缩处理

$$\begin{cases} CV_0 = IV, IV \text{为预设的初始值 (256 bit)} \\ CV_{i+1} = H_{SHA-1}(CV_i, M_i), 0 \leq i \leq L - 1 \\ H(M) = CV_L \end{cases}$$

- A=0x7380166f
- B=0x4914b2b9
- C=0x172442d7
- D=0xda8a0600
- E=0xa96f30bc
- F=0x163138aa
- G=0xe38dee4d
- H=0xb0fb0e4e



SM3压缩函数:

```
ABCDEFGH ← CV(i)
FOR j = 0 TO 63
    SS1 ← ((A <<< 12) + E + (Tj <<< j)) <<< 7;
    SS2 ← SS1 ⊕ (A <<< 12);
    TT1 ← FFj(A, B, C) + D + SS2 + W'j;
    TT2 ← GGi(E, F, G) + H + SS1 + Wj;
    D ← C
    C ← B <<< 9
    B ← A
    A ← TT1
    H ← G
    G ← F <<< 19;
    F ← E
    E ← P0(TT2)
ENDFOR
V(i+1) ← ABCDEFGH ⊕ V(i)
```

逻辑函数: (提供混淆作用)

$$FF_j(X,Y,Z) = \begin{cases} X \oplus Y \oplus Z & 0 \leq j \leq 15 \\ (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z) & 16 \leq j \leq 63 \end{cases}$$

$$GG_j(X,Y,Z) = \begin{cases} X \oplus Y \oplus Z & 0 \leq j \leq 15 \\ (X \wedge Y) \vee (\neg X \wedge Z) & 16 \leq j \leq 63 \end{cases}$$

置换函数: (提供扩散作用)

$$P_0(X) = X \oplus (X <<< 9) \oplus (X <<< 17)$$
$$P_1(X) = X \oplus (X <<< 15) \oplus (X <<< 23)$$

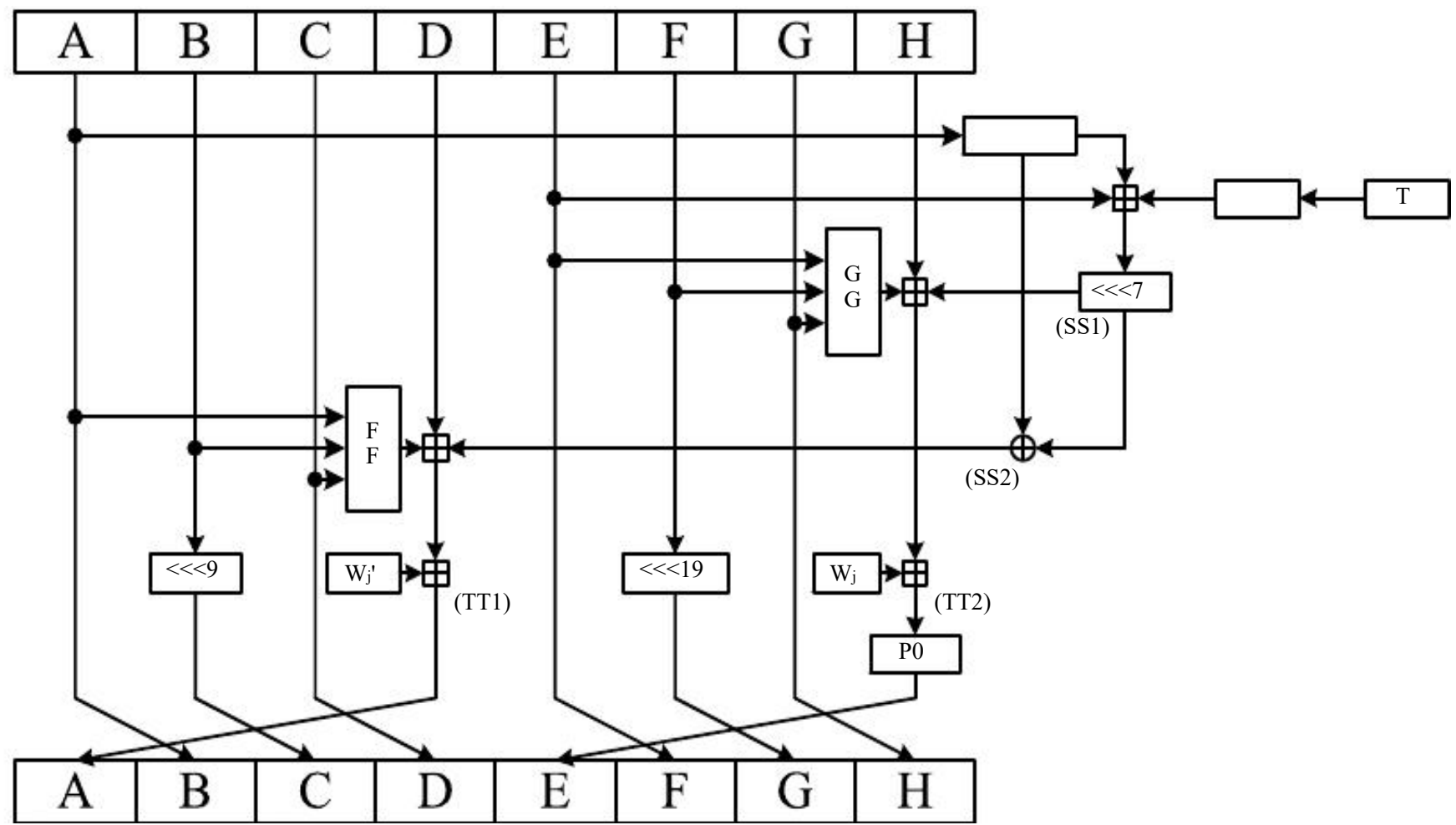
式中X,Y,Z为32位字, 符号a <<< n表示把a循环左移n位。

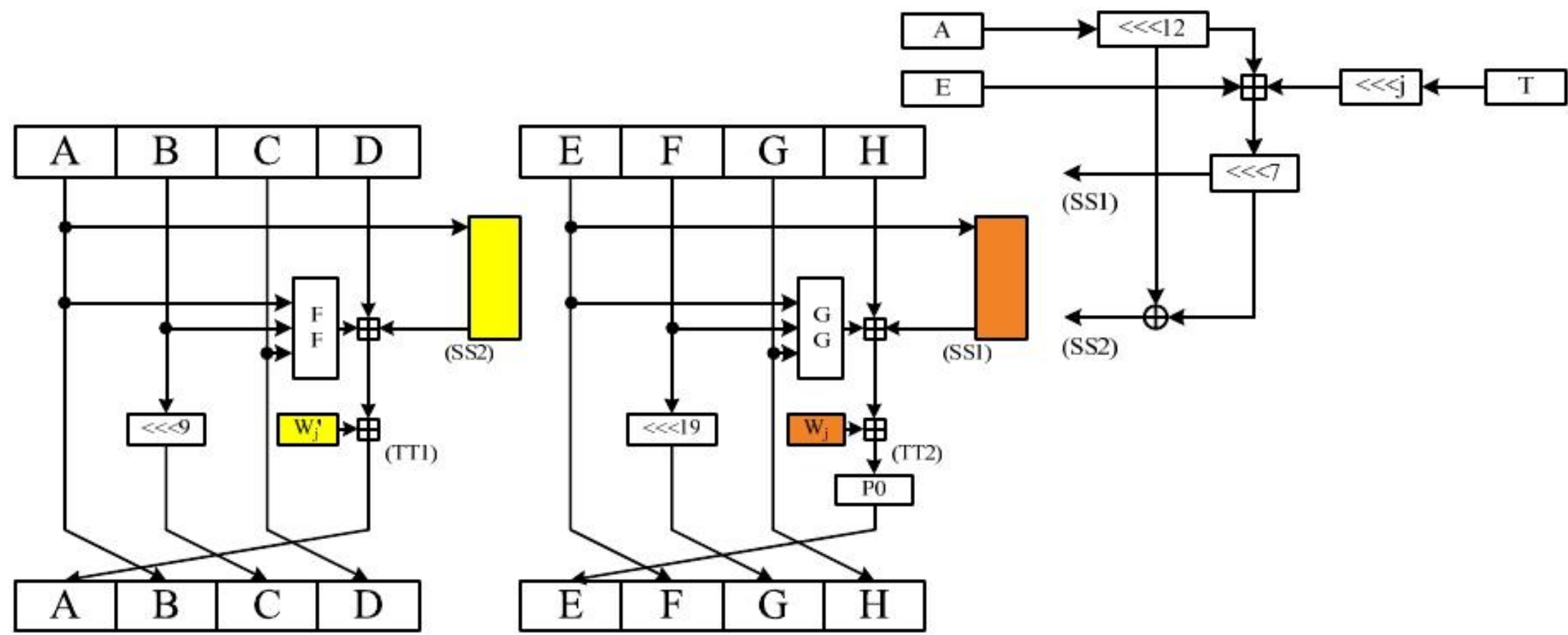
常量:

$$T_j = \begin{cases} 79cc45190 & 0 \leq j \leq 15 \\ 7a879d8a & 16 \leq j \leq 63 \end{cases}$$






512位数据分组扩展:

$$W_j = m_j \quad 0 \leq j \leq 15, \text{ 即512 bit分组划分成16个字}$$
$$W_j \leftarrow P_1(W_{j-16} \oplus W_{j-9} \oplus (W_{j-3} <<< 15)) \oplus W_{j-13} << < 7) \oplus W_{j-6} \quad 16 \leq j \leq 67$$
$$W'_j = W_j \oplus W_{j+4} \quad 0 \leq j \leq 63$$





### SM3哈希函数

-  我国商用密码杂凑算法标准
-  使用M-D结构
-  采用双路消息扩展输入
-  非对称Feistel结构
-  安全性只有经过实践检验，才能给出正确结论

# 章节安排

Outline



哈希函数简介

---



SHA-1哈希函数

---



SM3哈希函数

---



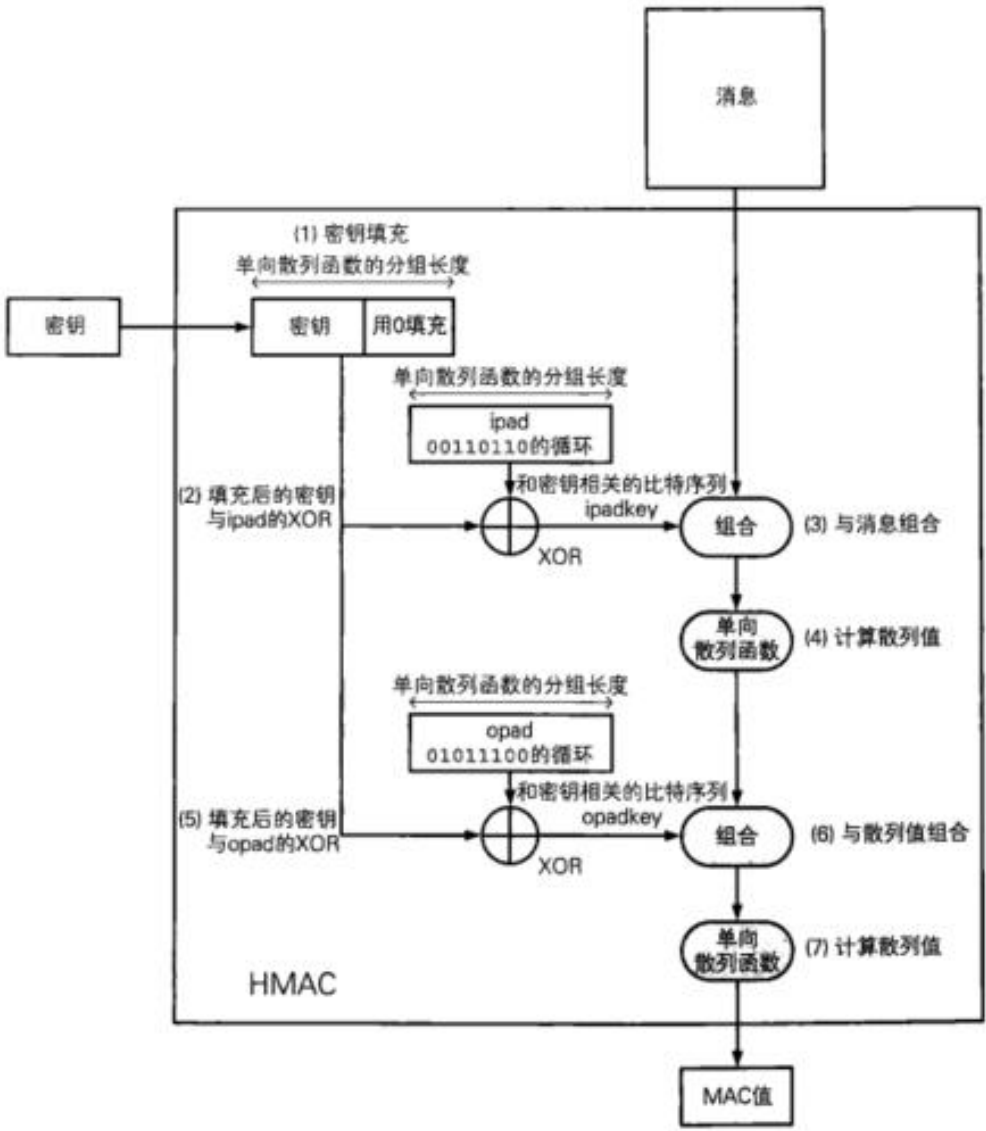
基于哈希的消息认证码HMAC

---

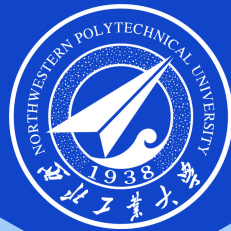
# 6.4 基于哈希的消息认证码HMAC

✎ HMAC是**密钥相关的哈希运算消息认证码**，是一种基于Hash函数和密钥进行消息认证的方法

- ✎ HMAC的构造：
- ✎ 基于分组密码算法构造
  - ✎ 基于Hash算法构造（HMAC）
- ✎ HMAC的作用：
- ✎ 消息完整性认证
  - ✎ 信源身份认证







感谢聆听!

THANK YOU FOR YOUR ATTENTION!