密码学

第七章 数字签名

网络空间安全学院
朱丹 戚明平
zhudan/mpqi@nwpu.edu.cn

知识回顾-安全哈希函数的应用

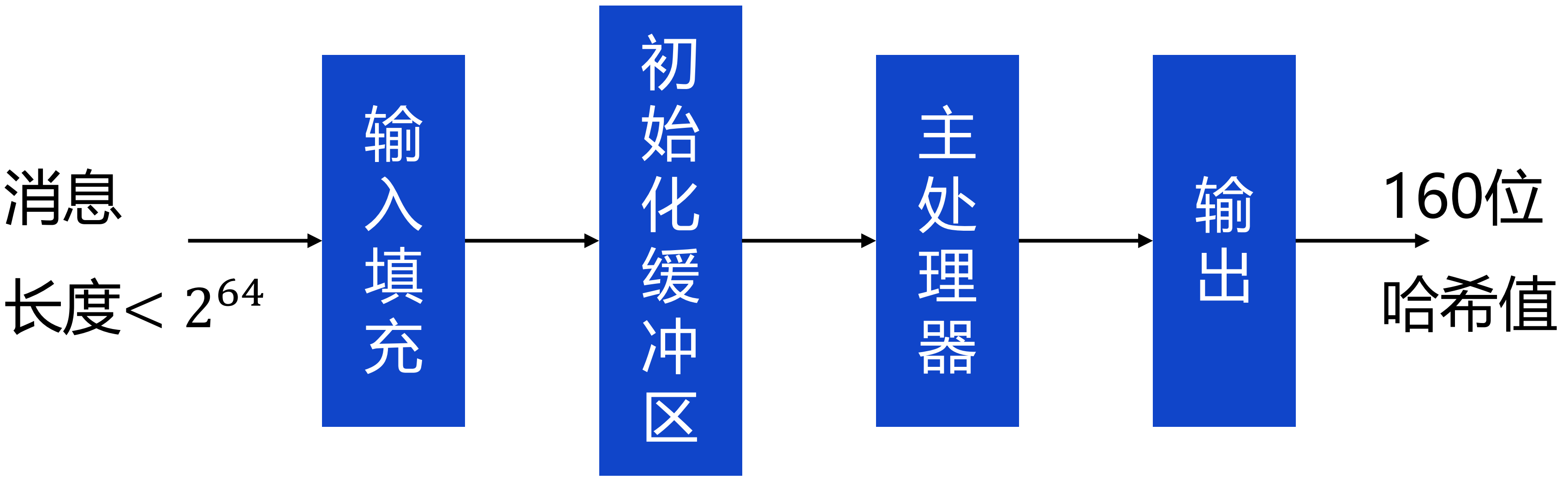
- 报文认证（方案a）：$A \to B: < M \parallel E(H(M), K) >$
- 报文认证（方案b）：$A \to B: < M \parallel H(M \parallel S) >$
- 报文认证和保密（方案a）：$A \to B: < E(M \parallel H(M), K) >$
- 报文认证和保密（方案b）：$A \to B: < E(M \parallel H(M \parallel S), K) >$
- 报文认证和数字签名：$A \to B: < M \parallel D(H(M), K_{dA}) >$
- 报文认证、数字签名和保密：$A \to B: < E(M \parallel D(H(M), K_{dA}), K) >$

知识回顾-SHA-1哈希函数

- 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

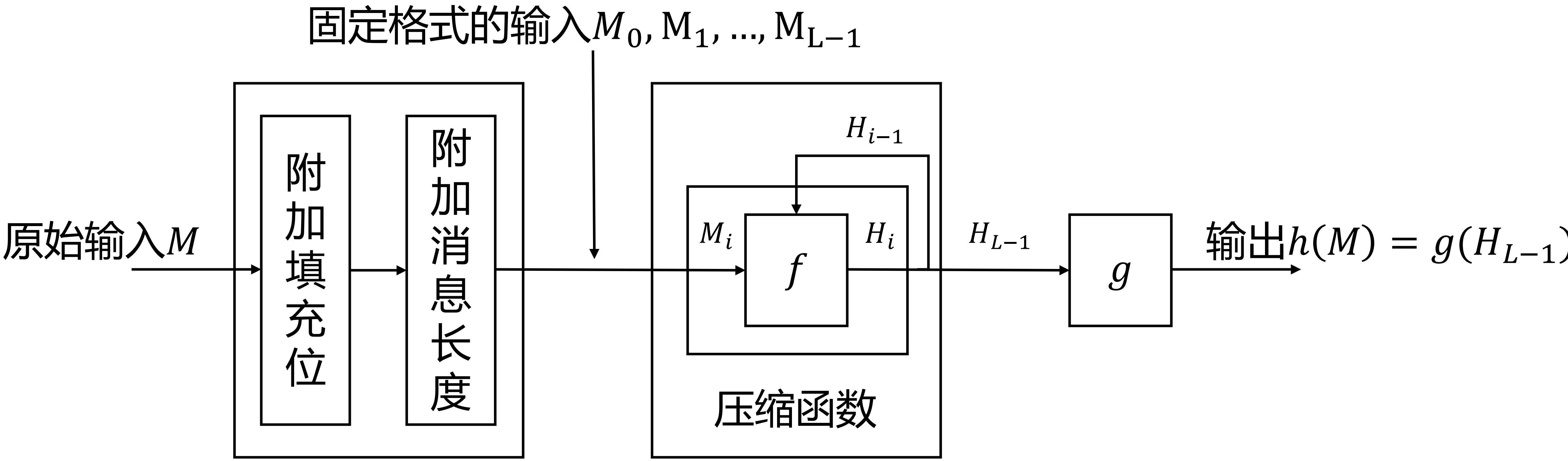
知识回顾-SHA-1哈希函数

- SHA-1的结构：采用Merkle提出的安全Hash模型



知识回顾-SHA-1哈希函数

- SHA-1算法



知识回顾-SHA-1哈希函数

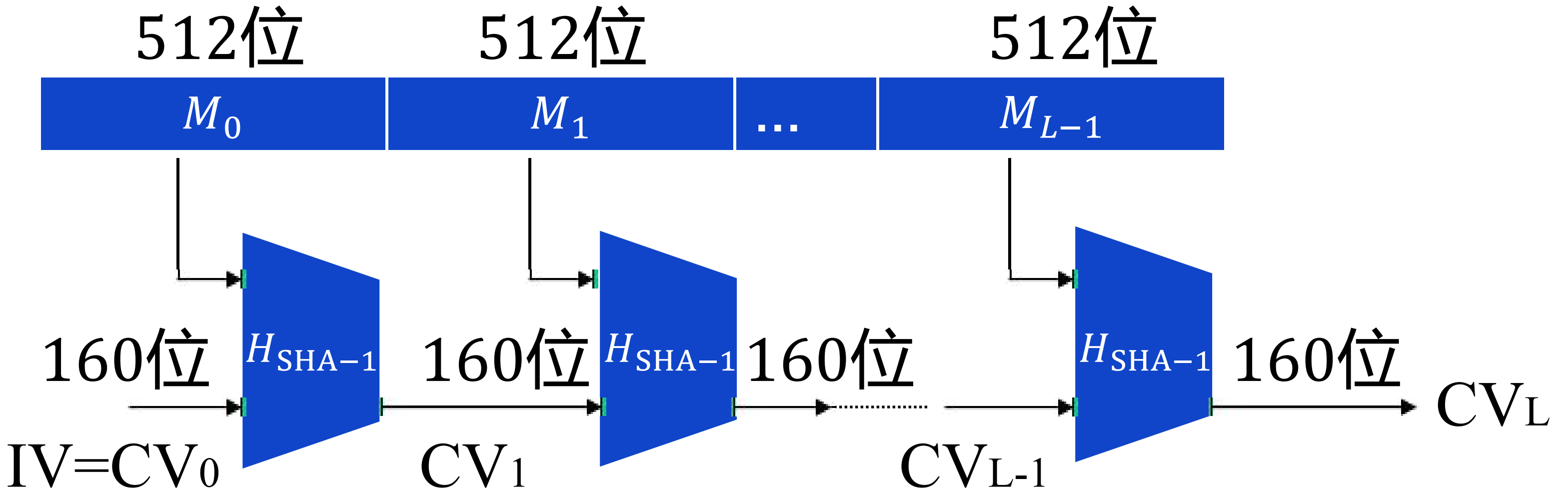
- SHA-1算法的输入填充
  - 目的是使填充后的报文长度满足：

$$长度=448\ mod\ 512$$

    - 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

知识回顾-SHA-1哈希函数

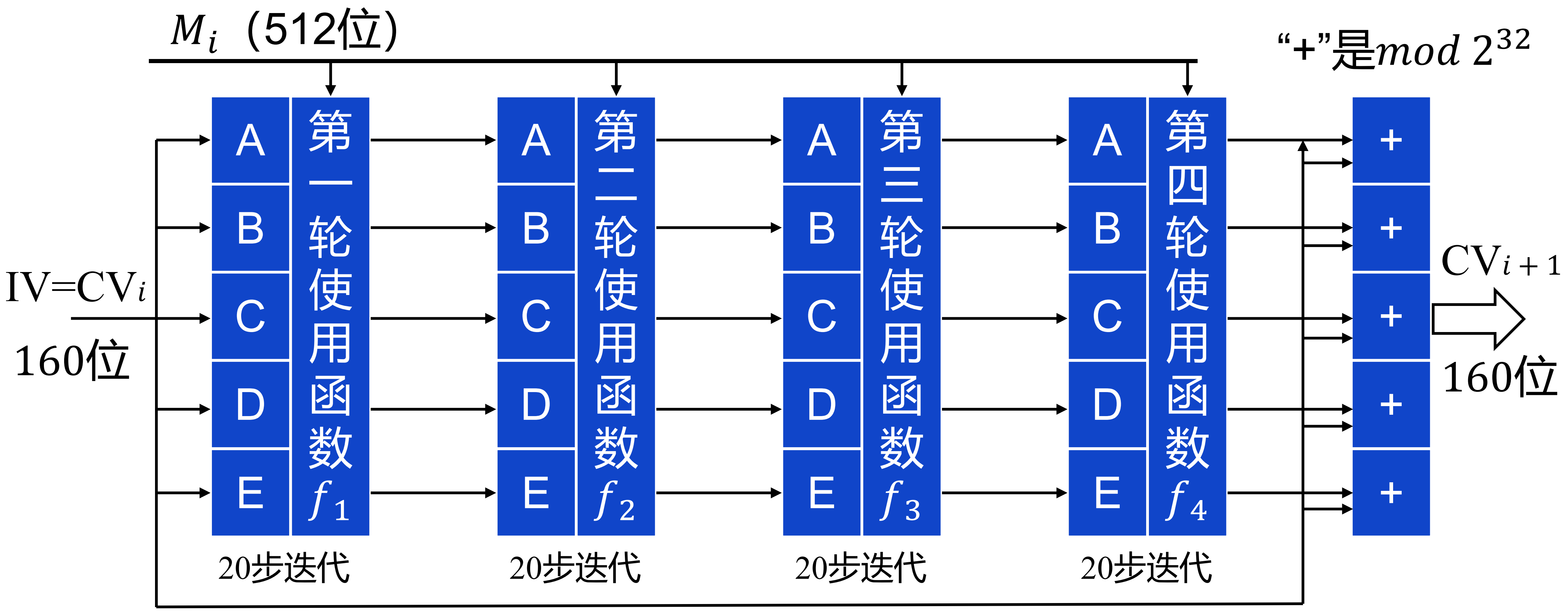
- SHA-1算法的主处理
  - 主处理是SHA-1 HASH函数的核心
  - 每次处理一个512位的分组，链接迭代处理（填充后报文）的所有L个分组数



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

知识回顾-SHA-1哈希函数

- SHA-1算法的主处理



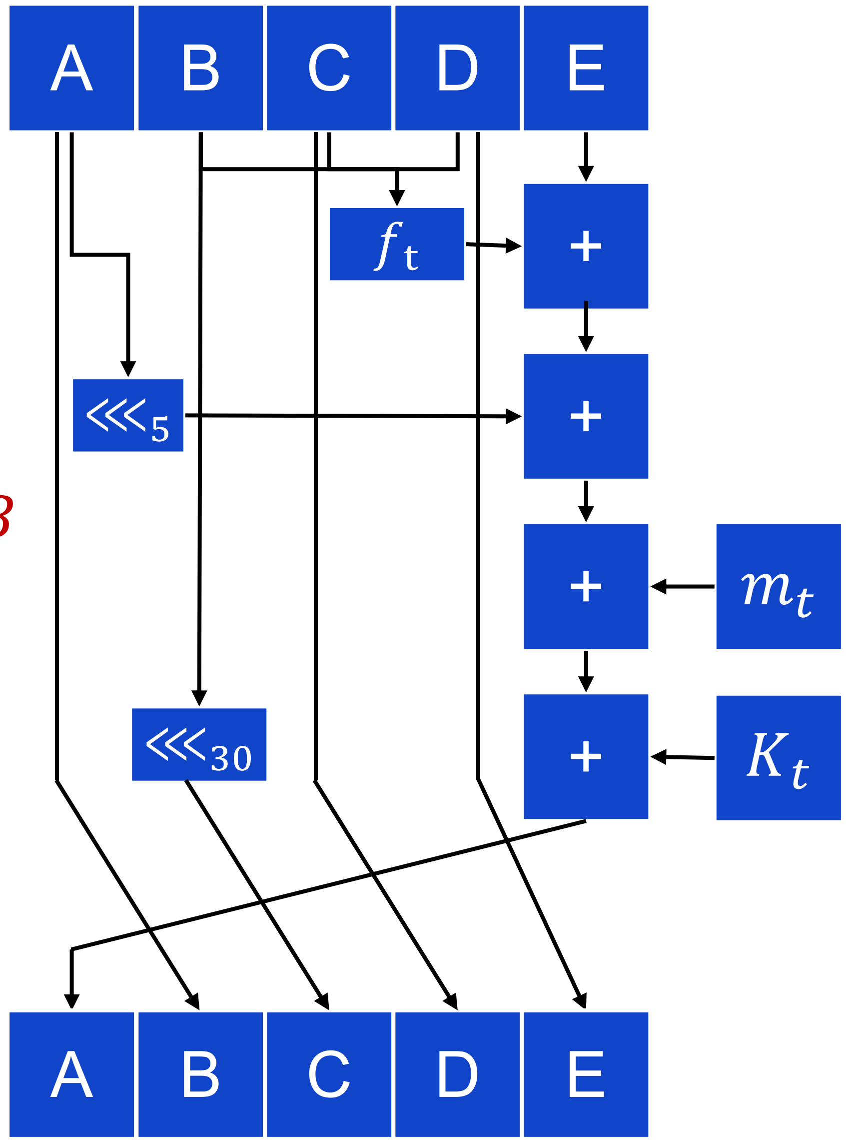
知识回顾-SHA-1哈希函数

- 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)

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

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

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



知识回顾-SHA-1哈希函数

- SHA-1算法的主处理
  - SHA-1算法压缩函数中的单步操作
    - 常量：$K_t = 0x5A827999 \quad 0 \leq t \leq 19$

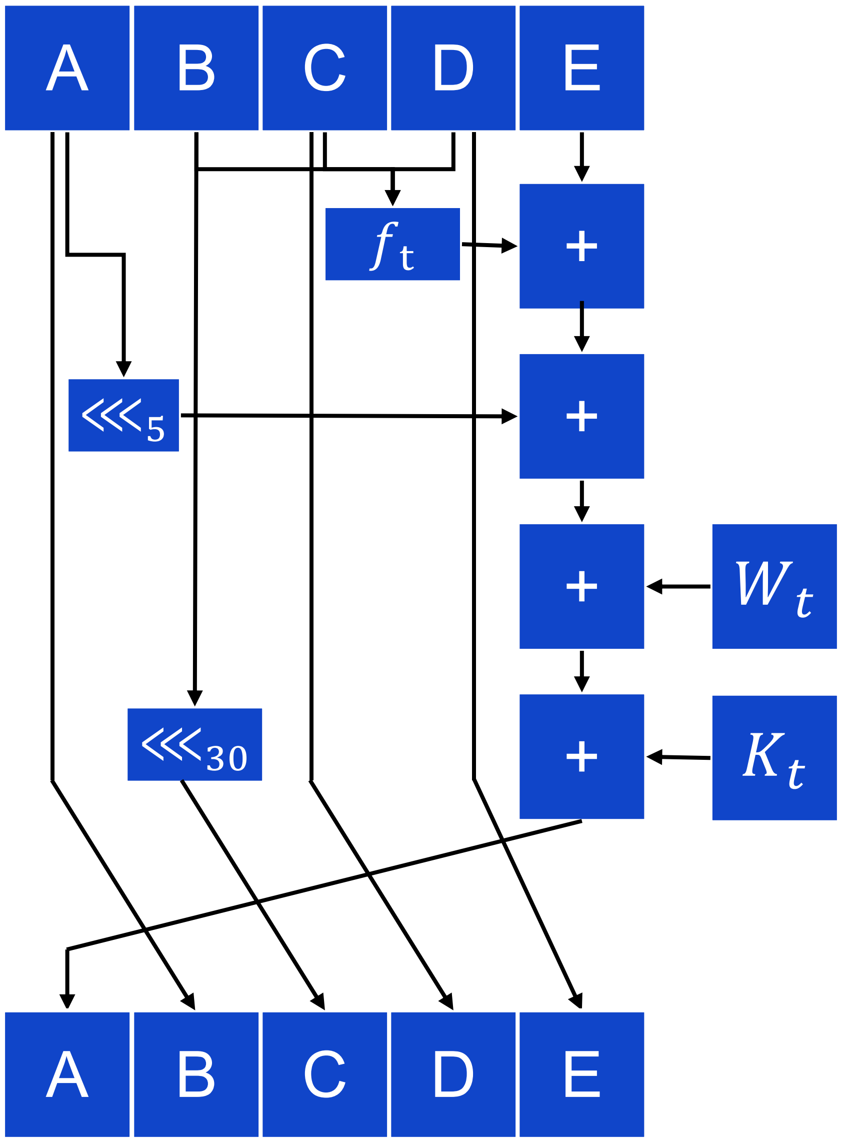
      $K_t = 0x6ED9EBA1 \quad 20 \leq t \leq 39$

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

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

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



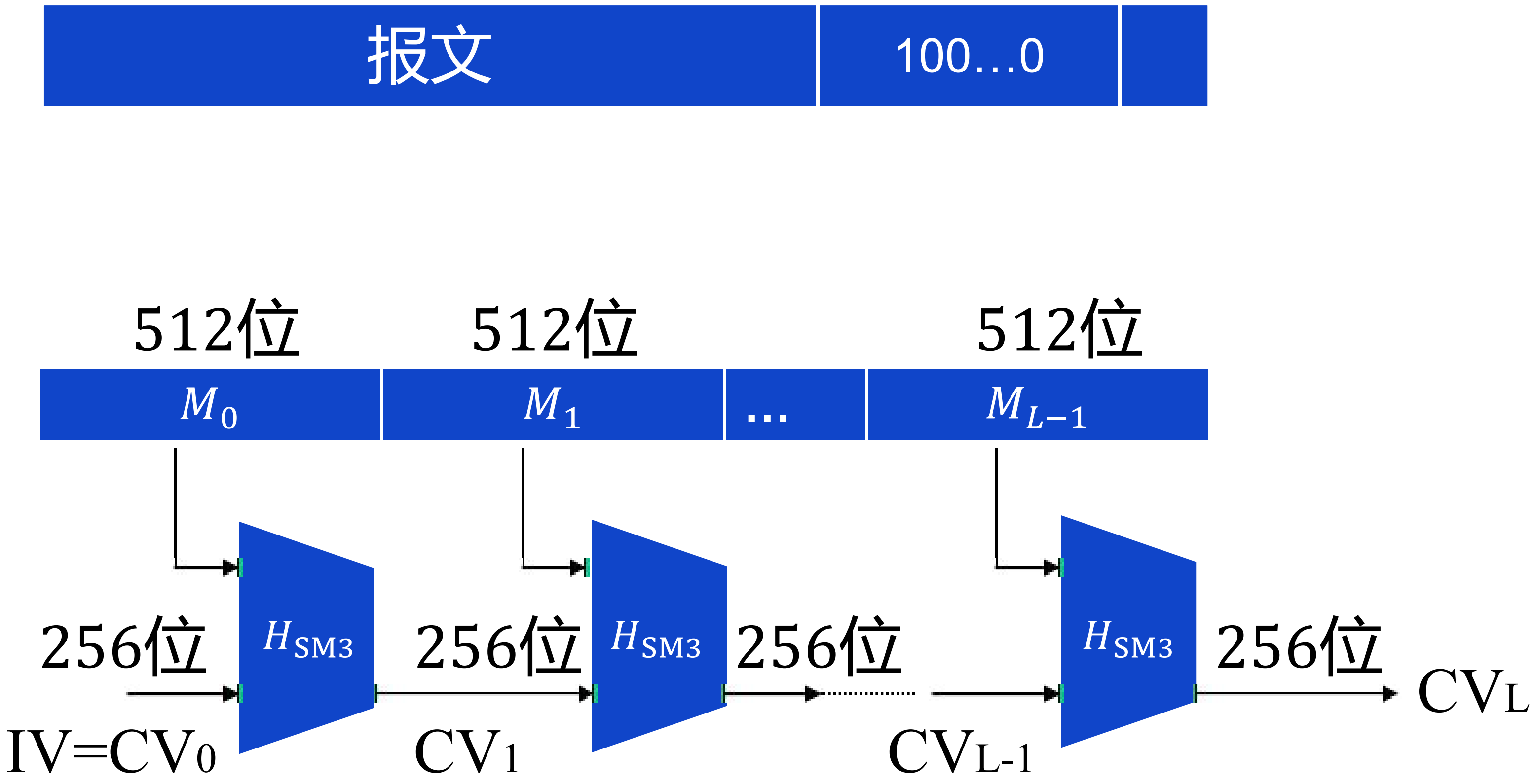
知识回顾-SM3哈希函数

- 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

知识回顾-SM3哈希函数

- SM3哈希函数的“压缩函数”＋“迭代结构”
  - 与SHA-1哈希函数相同



知识回顾-SM3哈希函数

SM3压缩函数:

$$
\begin{aligned}
& ABCDEFGH \leftarrow CV^{(i)} \\
& FOR\ j = 0\ \ TO\ \ 63 \\
& \quad SS1 \leftarrow \left(\left((A <<< 12) + E + \left(T_j <<< j\right)\right)\right) <<< 7; \\
& \quad SS2 \leftarrow SS1 \oplus (A <<< 12); \\
& \quad TT1 \leftarrow FF_j(A, B, C) + D + SS2 + W'_j \\
& \quad TT2 \leftarrow GG_i(E, F, G) + H + SS1 + W_j \\
& \quad D \leftarrow C \\
& \quad C \leftarrow B <<< 9 \\
& \quad B \leftarrow A \\
& \quad A \leftarrow TT_1 \\
& \quad H \leftarrow G \\
& \quad G \leftarrow F <<< 19: \\
& \quad F \leftarrow E \\
& \quad E \leftarrow P_0(TT2) \\
& ENDFOR \\
& V^{(i+1)} \leftarrow ABCDEFGH \oplus V^{(i)}
\end{aligned}
$$

逻辑函数: （提供混淆作用）

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

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

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:

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

512位数据分组扩展:

$W_j = m_j \qquad 0 \le j \le 15$，即512 bit分组划分成16个字

$W_j \leftarrow P_1(W_{j-16} \oplus W_{j-9} \oplus (W_{j-3} <<< 15)) \oplus W_{j-13} <<< 7) \oplus W_{j-6}$

$16 \le j \le 67$

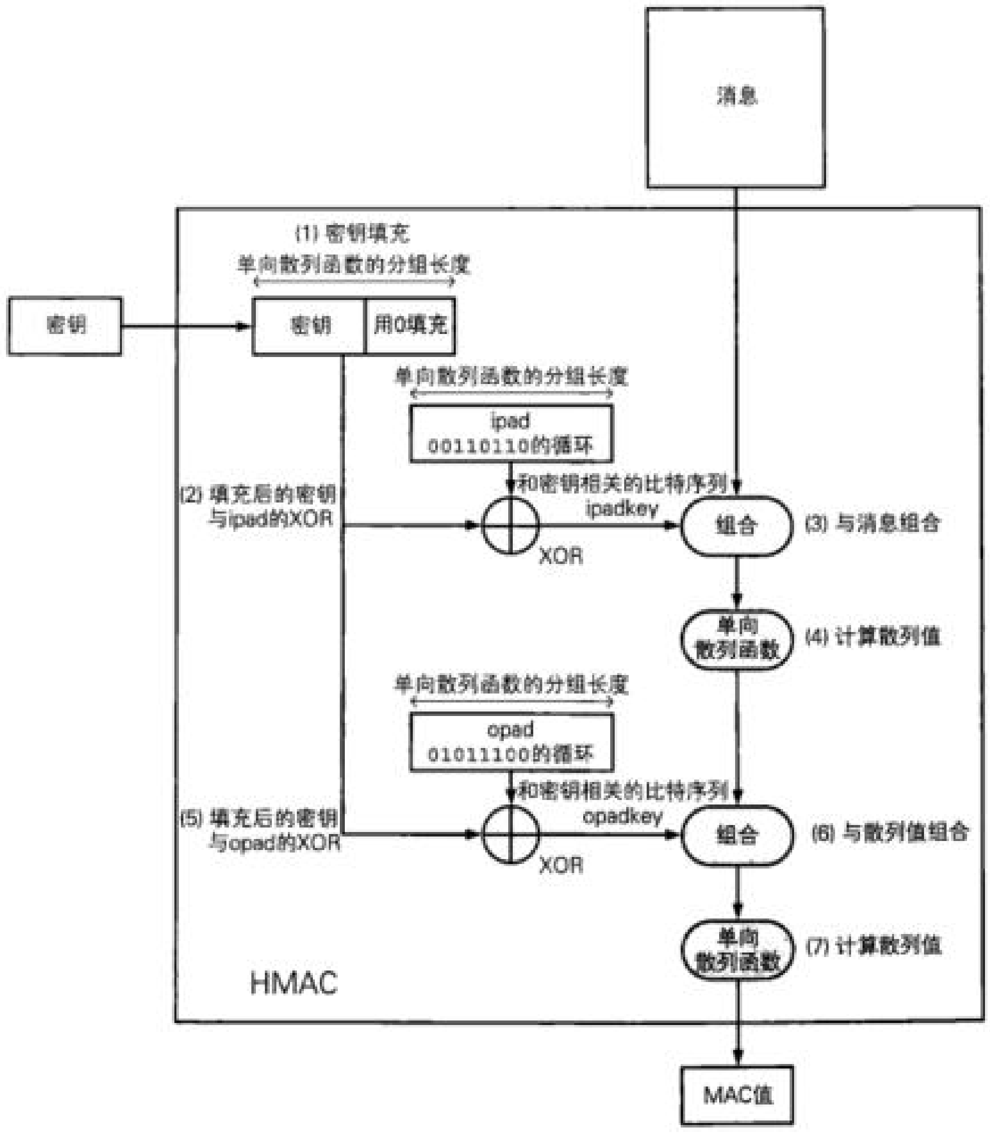
$W'_j = W_j \oplus W_{j+4} \qquad 0 \le j \le 63$

知识回顾-SM3哈希函数

- 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

知识回顾-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



章节安排

Outline

章节安排

Outline

7.1 数字签名的基本概念

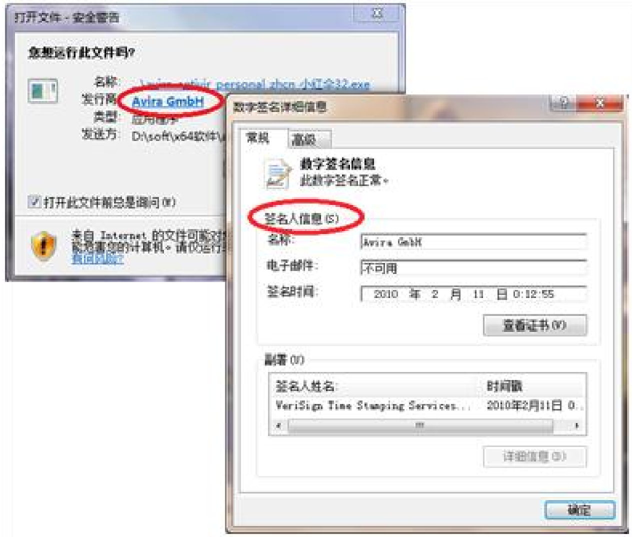
- 在人们的工作和生活中，许多事物的处理需要当事人签名。
- 签名起到确认、核准、生效和负责任等多种作用
- 签名是证明当事者的身份和数据真实性的一种信息
- 签名可以用不同的形式来表示

7.1 数字签名的基本概念

- 在传统的以书面文件为基础的事物处理中，采用书面签名的形式：手签、印章、手印等
- 书面签名得到司法部门的支持
- 在以计算机文件为基础的现代事物处理中，应采用电子数字形式的签名，即数字签名（Digital Signature）
- 数字签名已得到中国和其它一些国家的法律支持

7.1 数字签名的基本概念

- 一种完善的签名应满足以下四个条件：
  - 签名与文件具有绑定性
  - 签名者事后不能否认自己的签名
  - 任何其他人不能伪造签名
  - 如果当事双方关于签名真伪发生争执，能够在仲裁者面前通过验证确认其真伪



7.1 数字签名的基本概念

- 多基于公钥密码算法实现数字签名
- 数字签名的形式是多样的：
  - 通用签名、仲裁签名、盲签名、群签名、门限签名、代理签名等
  - 1994年，美国政府正式颁布了美国数字签名标准 DSS (Digital Signature Standard)
  - 1995 年，我国也制定了自己的数字签名标准（GB15851 - 1995）
  - 2004年，我国颁布了《中华人民共和国电子签名法》，签名可以用不同的形式来表示

章节安排

Outline

7.2 数字签名的模型

- 一个数字签名体制包括两个方面的处理：
  - 生成签名
  - 验证签名
- 设生成签名的算法为$\boldsymbol{SIG}$，产生签名的密钥为K_d，被签名的数据为M，产生的签名信息为S，则有

$$S = \boldsymbol{SIG}(M, K_d)$$

- 设验证签名的算法为$\boldsymbol{VER}$，验证签名的密钥为K_e，用$\boldsymbol{VER}$对签名S进行验证，可鉴别S的真假。即

$$\boldsymbol{VER}(S, K_e) = \begin{cases} \text{真, 当验证结果符合判定准则} \\ \text{假, 当验证结果不符合判定准则} \end{cases}$$

7.2 数字签名的模型

- 签名函数必须满足以下条件，否则文件内容及签名被篡改或冒充时均无法发现：

  ① 当$M \neq M'$时，有$\boldsymbol{SIG}(M, K_d) \neq \boldsymbol{SIG}(M', K_d)$

  ✓ 条件①要求签名S至少和被签名的数据M一样长（一一映射关系）。当M太长时，应用很不方便

  ✓ 将条件①改为：虽然当$M \neq M'$时，存在$S = S'$，但对于给定的M或S，要找出具有相同签名的M'在计算上是不可能的

- 签名函数必须满足以下条件，否则文件内容及签名被篡改或冒充时均无法发现：

  ② 签名S只能由签名者产生，否则别人便可伪造，于是签名者也就可以抵赖

  ③ 收信者可以验证签名S的真伪。这使得当签名为假是收信者不必上当

  ④ 签名者也应有办法鉴别收信者所出示的签名是否是自己的签名。这就给签名者以自卫的能力

章节安排

Outline

7.3(1) 利用公钥密码实现数字签名

- **利用公钥密码实现数字签名的一般方法：**
  - 对于一个公钥密码，如果满足

$$E(D(M, K_d), K_e) = M$$

  则可确保数据的真实性。

- 凡是能够确保数据真实性的公钥密码都可用来实现数字签名。例如：
  - RSA密码
  - ElGamal密码
  - 椭圆曲线密码
- 但是一个数字签名方案不一定能够满足上式。

7.3(1) 利用公钥密码实现数字签名

- 利用公钥密码实现数字签名的一般方法：
- 为了实施数字签名，应成立管理机构
  - 制定规章制度
  - 统一技术标准
  - 用户登记注册
  - 纠纷的仲裁
  - 其他

7.3(1) 利用公钥密码实现数字签名

- **利用公钥密码实现数字签名的一般方法:**
- 数字签名—消息验证过程:

签名通信协议: $A \xrightarrow{M} B$

① A用自己的私钥K_{d_A}对数据M进行签名:

$$S_A = \boldsymbol{SIG}(M, K_{d_A})$$

② 如果不需要保密，则A直接将S_A发送给用户B

③ 如果需要保密，则A用B的公钥K_{e_B}对S_A加密，得到密文C:

$$C = E(S_A, K_{e_B})$$

④ 最后，A将C发送给B，并将S_A或C留底

7.3(1) 利用公钥密码实现数字签名

⑤ B收到后，若是不保密通信，则用A的公钥K_{e_A}对签名进行验证：

$$\boldsymbol{VER}(S_A, K_{e_A}) = \boldsymbol{VER}(\boldsymbol{SIG}(M, K_{d_A}), K_{e_A}) \in \{0,1\}$$

⑥ 若是保密通信，则B先用自己的私钥K_{d_B}对C解密，然后再用A的公钥K_{e_A}对签名进行验证：

$$D(C, K_{d_B}) = D(E(S_A, K_{e_B}), K_{d_B}) = S_A$$

$$\boldsymbol{VER}(S_A, K_{e_A}) = \boldsymbol{VER}(\boldsymbol{SIG}(M, K_{d_A}), K_{e_A}) \in \{0,1\}$$

⑦ 如果验证结果为真，则说明S_A是A的签名，否则S_A不是A的签名

⑧ B对收到的C或S_A留底

7.3(1) 利用公钥密码实现数字签名

- **利用公钥密码实现数字签名的一般方法：**
- 签名通信协议安全性分析：
  - 因为只有A才拥有K_{d_A}，而且由公开的K_{e_A}在计算上不能求出保密的私钥K_{d_A}
  - 签名的操作只有A才能进行，其他任何人都不能进行
  - K_{d_A}就相当于A的印章或指纹，而S_A就是A对M的签名
  - 对此A不能抵赖，其他任何人不能伪造
  - 事后如果A和B关于签名的真伪发生争执，则他们应向公正的仲裁者出示留底的签名数据，由仲裁者当众验证签名，解决纠纷

7.3(1) 利用公钥密码实现数字签名

- **利用公钥密码实现数字签名的一般方法：**
- 签名通信协议的问题：
  - 验证签名的过程就是恢复明文的过程。而B事先并不知道明文M，否则就用不着通信了。那么B怎样判定恢复出的M是否正确呢？
  - 怎样阻止B或A用A以前发给B的签名数据，或用A发给其他人的签名数据来冒充当前A发给B的签名数据呢？
  - 仅仅靠签名本身并不能解决这些问题。

7.3(1) 利用公钥密码实现数字签名

- **利用公钥密码实现数字签名的一般方法：**
- 解决问题的一种办法：
  - 合理设计明文的数据格式：

发方标识	收方标识	报文序号	时间	数据	纠错码

$$M =< A,\ B, I, T, DATA, CRC >$$

  - A将$< H, SIG(M, K_{d_A}) >$为最终报文发送给B，其中$H =< A, B, I, T >$为明文。
  - 只要用A的公钥验证签名并恢复出正确的附加信息$H =< A, B, I, T >$，便可断定明文M是否正确。
  - 设附加信息H的二进制长度为L，则错判概率$p^e \leq 2^{-L}$。

7.3(1) 利用公钥密码实现数字签名

- **利用公钥密码实现数字签名的一般方法：**
- 改进：
  - 保留上述方法的报头处理，数据签名改为：对$Hash(M)$ 签名，而不直接对M签名。

数据M	$Hash(M)$

$$S = \boldsymbol{SIG}\left(Hash(M), K_{d_A}\right)$$

传输格式：$< M, S >$

数据M	签名S

  - 设收到的数据为$< M', S' >$，仅当$Hash(M') = E(S', K_{e_A})$且报头是正确时，判断M是正确的。

- **利用RSA密码实现数字签名：**
  - 对于RSA密码：

$$D\big(E(M)\big) = (M^e)^d = M^{ed} = (M^d)^e = E\big(D(M)\big) \bmod n$$

  - 所以RSA密码可同时确保数据的机密性和真实性
  - 利用RSA密码可以同时实现数据加密和数字签名

7.3(2) 利用公钥密码实现数字签名

- **利用RSA密码实现数字签名：**
  - 设M为明文，$K_{e_A} =< e, n >$是A的公钥
  - $K_{d_A} =< p, q, \varphi(n), d >$是A的私钥
- 则A对M的签名过程是：
  - $S_A = D(M, K_{d_A}) = M^d \bmod n$
  - S_A便是A对M的签名
- 验证签名的过程是：
  - $E(S_A, K_{e_A}) = (M^d)^e \bmod n = M$

- **对RSA数字签名的攻击**
- 一般攻击：
  - 因为e和n是A的公开密钥，所以任何人都可以获得并使用e和n。攻击者可随意选择一个数据Y，并用A的公钥计算统一技术标准

$$X = Y^e \bmod n$$

  - 因为$Y = X^d \bmod n$，于是可以用Y伪造A的签名。因为Y是A对X的一个有效签名

这样的X往往无正确语义，因此这种攻击在实际上有效性不大

7.3(2) 利用公钥密码实现数字签名

- **对RSA数字签名的攻击**
- 利用已有的签名进行攻击：
  - 攻击者选择随机数据 M_3，且$M_3 = M_1 M_2 \bmod n$。
  - 攻击者设法让A对 M_1和M_2签名：

$$S_1 = M_1^d \bmod n\,,\quad S_2 = M_2^d \bmod n$$

  - 于是可以由S_1和S_2计算出A对M_3的签名：

$$S_1 S_2 = M_1^d M_2^d \bmod n = M_3^d \bmod n = S_3$$

对策：A不直接对数据*M*签名，而是对**HASH(*M*)**签名

- **对RSA数字签名的攻击**
- 利用已有的签名进行攻击:

$$S_1 = \left(HASH(M_1)\right)^d \bmod n$$

$$S_2 = \left(HASH(M_2)\right)^d \bmod n$$

- 而:

$$\left(HASH(M_1)\right)^d \left(HASH(M_2)\right)^d \neq \left(HASH(M_1 M_2)\right)^d \bmod n$$

- 所以, $S_3 \neq S_1 S_2$

于是不能由*S*1和*S*2计算出A对*M*3的签名

7.3(2) 利用公钥密码实现数字签名

- **对RSA数字签名的攻击**
- 攻击签名获得**明文**:
  - 攻击者截获C，$C = M^e \bmod n$
  - 攻击者选择小的随机数r，计算:

$$x = r^e \bmod n,\ y = xC \bmod n,\ t = r^{-1} \bmod n$$

  - 攻击者让A对y签名，于是攻击者又获得:

$$S = y^d \bmod n$$

  - 攻击者计算$tS = r^{-1}y^d = r^{-1}(xC)^d = C^d = M \bmod n$

对策：**A不直接对数据*M*签名，而是对HASH(*M*)签名**

- **对RSA数字签名的攻击**
- 对先加密后签名方案的攻击：
  - A采用先加密后签名的方案发送M给B：

$$S = ((M)^{e_B} \, mod \, n_B)^{d_A} \, mod \, n_A$$

  - 如果B不诚实，则他可以找到 M_1满足:

$$(M_1)^{x_{e_B}} = M^{e_B} mod \, n_B$$

  - 则B可以重新公开密钥x_{e_B}, 并宣称他收到的是 M1

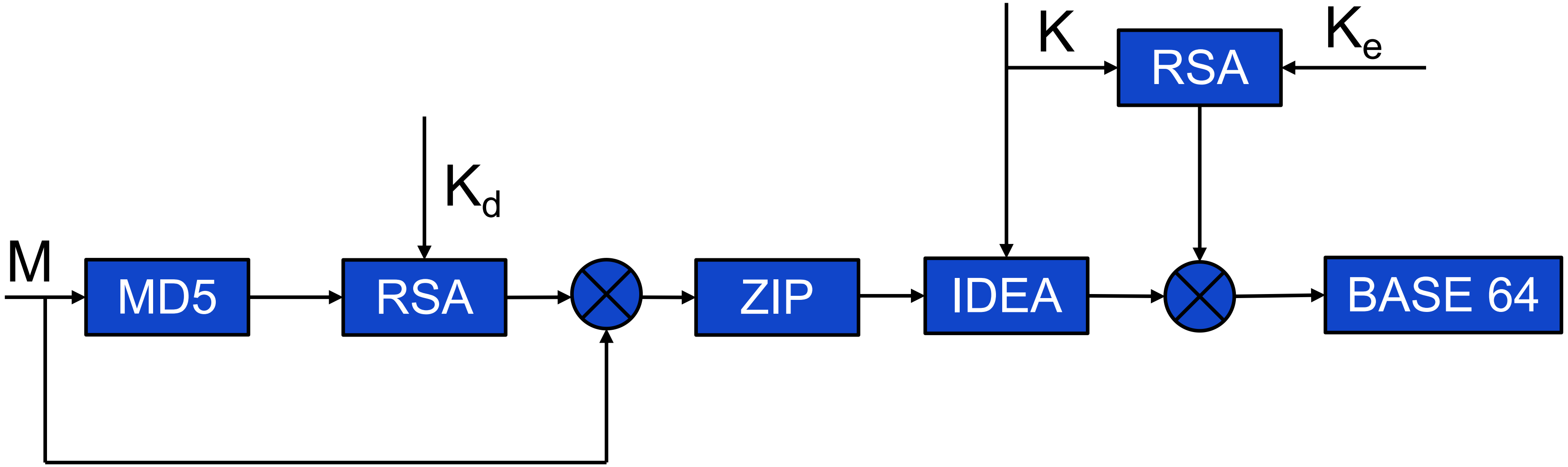
对策：(1) A在发送数据中加入时间戳；(2)对**HASH(*M*)**签名; (3) 先签名再加密

- **对RSA数字签名的攻击**
- 结论:
  - 不直接对数据M签名，而是对HASH(M)签名
  - 使用时间戳
  - 对于同时确保机密性和真实性的通信，应当先签名后加密

7.3(2) 利用公钥密码实现数字签名

- **RSA数字签名的应用**
- RSA数字签名的应用：PGP (Pretty Good Privacy)
  - 数据M经MD5处理，得到MD5(M)
  - 利用RSA对HASH(M)签名,得到M的签名S
  - 使用ZIP对$< M, S >$压缩
  - 再用IDEA对压缩数据加密：IDEA($ZIP(M, S)$)
  - 用RSA对IDEA的密钥加密：RSA(k)
  - 形成数据：$<$ IDEA($ZIP(M, S)$), RSA(k) $>$
  - 将数据转换成ASCII码

- **RSA数字签名的应用**
- RSA数字签名的应用：PGP (Pretty Good Privacy)



- **利用ElGamal密码实现数字签名：**
  - 密钥选择：选p是一个大素数，$p-1$有大素因子，a是模p的一个本原元，将p和a公开作为密码基础参数；用户随机地选择一个整数x（$1 \le x \le p-1$）作为私有的解密密钥；计算$y = a^x \bmod p$，取y作为公开的加密密钥。
  - 产生签名：设明文为M，$0 \le M \le p-1$，签名过程如下

    ① 用户A随机地选择一个整数k, $1 < k < p-1$，且$(k, p-1) = 1$

    ② 计算$r = a^k \bmod p$

    ③ 计算$s = (M - xr)k^{-1} \bmod p-1$

    ④ 取(r, s)作为M的签名，并以$< M, r, s >$的形式发给用户B

7.3(2) 利用公钥密码实现数字签名

- **利用ElGamal密码实现数字签名：**
  - 用户B接收$< M, r, s >$，用A的公钥验证

$$a^M = y^r r^s \ mod\ p$$

是否成立，若成立则签名为真，否则签名为假。

  - 签名的可验证性证明如下：

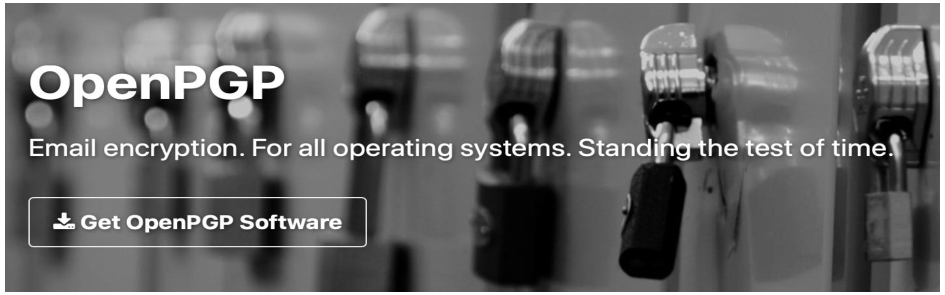
因为$s = (M - xr)k^{-1} \ mod\ p - 1$

所以$M = sk + xr \ mod\ p - 1$

所以$a^M = a^{sk+xr} = a^{sk}a^{xr} = y^r r^s \ mod\ p$，故签名可以验证。

课后阅读

- PGP - http://www.pgp.cn/index.htm
- OpenPGP - https://www.openpgp.org
- GnuPG - https://www.gnupg.org



感谢聆听!

THANK YOU FOR YOUR ATTENTION!