山东大学
SHANDONG UNIVERSITY

网络与大数据安全

2 - Cryptography

李琨
Email: kunli@sdu.edu.cn

学无止境 气有浩然

山东大学 SHANDONG UNIVERSITY

目录

CONTENTS

1.密码学发展史

2.古典密码学

3.DES与AES

4.密钥交换与公钥密码学

5.RSA与ECC

6.数字签名与MAC

回顾

- 密码学发展史：
  - 原始符号、宗教符号、消息隐藏
  - 古典密码、现代密码、未来密码
- 古典密码
  - 代换加密与仿射密码
  - 置换加密
  - 流密码
  - 分组密码
- DES与AES
  - SPN结构
  - Feistel结构
  - DES算法
  - AES算法

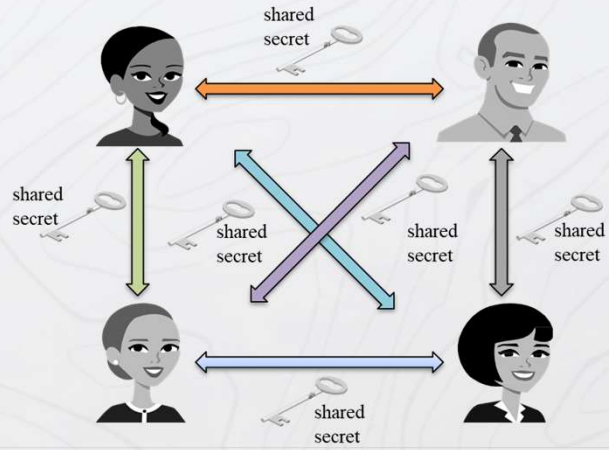
混淆：将密文与密钥之间的统计关系变得尽可能复杂，使得对手即使获取了关于密文的一些统计特性，也无法推测密钥

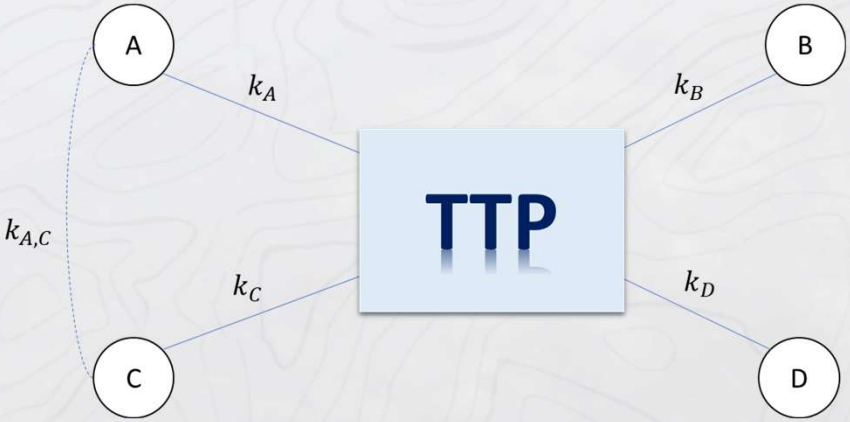
扩散：让明文中的每一位影响密文中的许多位，或者说让密文中的每一位受明文中的许多位的影响．这样可以隐蔽明文的统计特性

密钥交换与公钥密码学

- 问题：为了实现加密与解密，N个用户如何交换密码
  - 方法1：**通过事先共享密钥来解决。**事先用安全的方式将密钥交给对方，这称为密钥的**事先共享**。

    局限性：有时候难以找到一种安全的方式将密钥交给对方，此外，在人数很多的情况下，通信所需要的密钥数量也会增大，这是不现实的。
  - 方法2：通过密钥分配中心(KDC)来解决。当需要进行加密通信时，密钥分配中心会生成一个通信密钥，每个人只要和密钥分配中心事先共享密钥即可。密钥分配中心拥有所有人的密钥，而每个人则拥有自己的密钥。

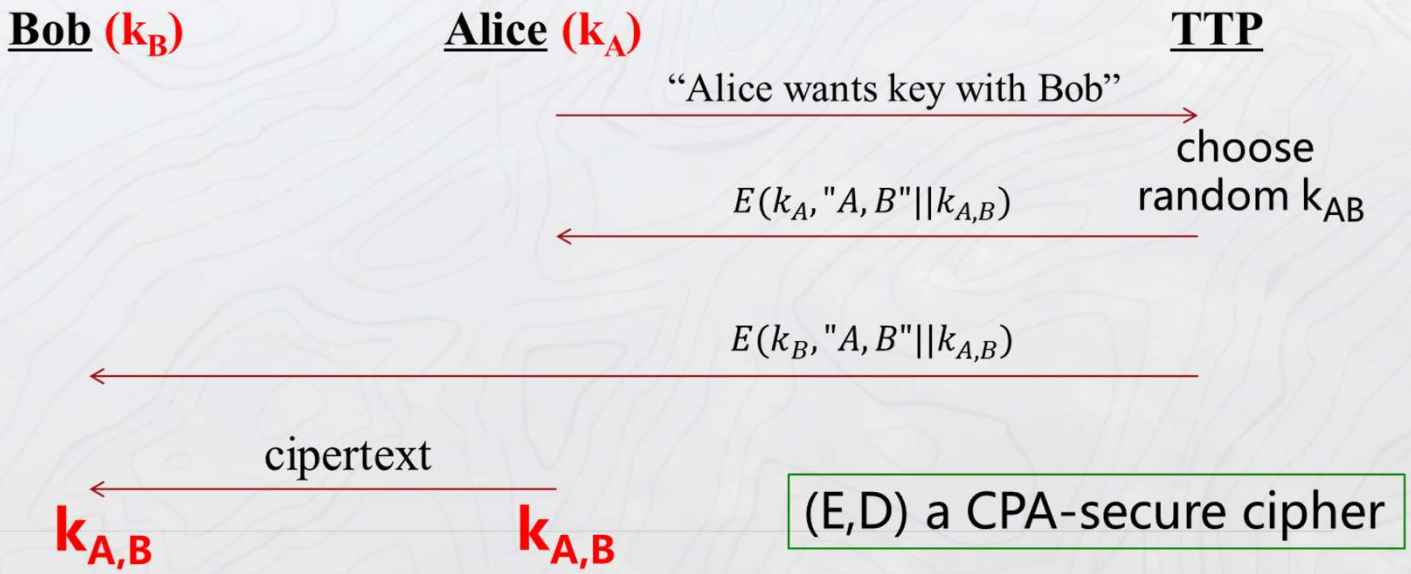




- 问题：为了实现加密与解密，N个用户如何交换密码——TTP

以Alice向Bob发送加密邮件为例

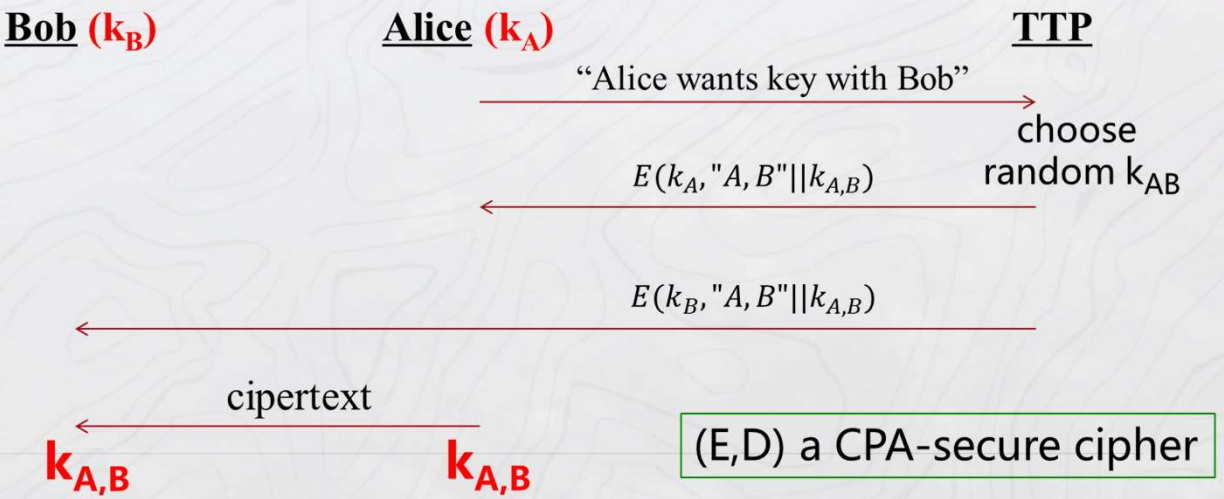
1. Alice向密钥分配中心发出希望与Bob进行通信的请求。
2. 密钥分配中心通过伪随机数生成器生成一个**会话密钥**，这个密钥是供Alice与Bob在本次通信中使用的临时密钥。
3. 密钥分配中心从数据库中取出Alice的密钥和Bob的密钥。
4. 密钥分配中心分别用Alice和Bob的密钥对会话密钥进行加密，并分别发送给Alice和Bob。



- 问题：为了实现加密与解密，N个用户如何交换密码——TTP

以Alice向Bob发送加密邮件为例

5. Alice对来自密钥分配中心的会话密钥（已使用Alice的密钥加密）进行解密，得到会话密钥。
6. Alice用会话密钥对邮件进行加密，并将邮件发送给Bob。
7. Bob对来自密钥分配中心的会话密钥（已使用Bob的密钥加密）进行解密，得到会话密钥。
8. Bob用会话密钥对来自Alice的密文进行解密。
9. Alice和Bob删除会话密钥。



- 问题：为了实现加密与解密，N个用户如何交换密码——TTP

以Alice向Bob发送加密邮件为例

- 可能存在的攻击1：窃听

窃听者能获取$E(k_A, \text{“}A, B\text{”}||k_{A,B})$和$E(k_B, \text{“}A, B\text{”}||k_{A,B})$，但是无法获取$k_{A,B}$

- 可能存在的攻击2：重放

攻击者记录Alice发送给Bob的信息，尽管无法解密，但可以再次发送给Bob

例：该信息是银行交易请求，则Bob会误以为Alice发起了第二个请求

- 可能存在的攻击3：针对密钥分配中心的攻击，直接获取所有用户的密钥库

- 可能存在的攻击4：随着人数的增加，密钥分配中心的负荷也会增加。如果密钥分配中心计算机发生故障，全公司的加密通信就会瘫痪。

- 问题：为了实现加密与解密，N个用户如何交换密码——**Puzzles**

通过Puzzles实现：需要付出一定努力来求解的问题

Alice: prepare 2^{32} puzzles

- For $i = 1, \ldots, 2^{32}$ choose random $\boldsymbol{P_i} \in \{\mathbf{0},\mathbf{1}\}^{\mathbf{32}}$ and $\boldsymbol{x_i}, \boldsymbol{k_i} \in \{\mathbf{0},\mathbf{1}\}^{\mathbf{128}}$

  set $puzzle_i \leftarrow E(0^{96} \,||\, \boldsymbol{P_i}, \text{“}\boldsymbol{Puzzle\ \#\ x_i}\text{”} \,||\, \boldsymbol{k_i})$

- Send $\text{puzzle}_1, \ldots, \text{puzzle}_{232}$ to Bob

Bob: choose a random $puzzle_j$ and solve it. Obtain (x_j, k_j). O(n)

- Send x_j to Alice, save k_j

Alice: lookup puzzle with number x_j. Use k_j as shared secret

- 问题：为了实现加密与解密，N个用户如何交换密码——**Puzzles**

通过Puzzles实现：需要付出一定努力来求解的问题

例：$E(k,m)$ 是对称加密，其中$k \in \{0,1\}^{128}$

$\boldsymbol{puzzle(P)} = \boldsymbol{E(P,\text{“}message\text{”})}$ $where$ $P = 096 \parallel b_1 \dots b_{32}$

目标：通过尝试2^{32}种可能来寻找正确的P

Alice: prepare 2^{32} puzzles O(n)

- For $i = 1, \dots, 2^{32}$ choose random $\boldsymbol{P_i} \in \{\boldsymbol{0,1}\}^{\boldsymbol{32}}$ and $\boldsymbol{x_i}, \boldsymbol{k_i} \in \{\boldsymbol{0,1}\}^{\boldsymbol{128}}$

  set $puzzle_i \leftarrow E(\ 0^{96} \parallel \boldsymbol{P_i}, \text{“}\boldsymbol{Puzzle\ \#\ x_i}\text{”} \parallel \boldsymbol{k_i}\ )$

- Send puzzle$_1$, … , puzzle$_{232}$ to Bob

Bob: choose a random $puzzle_j$ and solve it. Obtain (x_j, k_j) . O(n)

- Send x_j to Alice, save k_j

Alice: lookup puzzle with number x_j . Use k_j as shared secret

Eavesdropper: Solve all puzzles. $O(n^2)$ (e.g. 2^{64} times) ⇨ 平方鸿沟

所谓求解谜题，实际上是逐一尝试所有可能，直到解出前几位是Puzzle的句子

- 问题：为了实现加密与解密，N个用户如何交换密码——DH协议

Diffie-Hellman协议

Diffie-Hellman Set-up

1. Choose a large prime p.
2. Choose an integer $g \in \{1,2,\ldots,p-1\}$.
3. Publish p and g.

Diffie-Hellman Key Exchange

Alice		Bob
choose $a = k_{pr,A} \in \{1,2,\ldots,p-1\}$		choose $b = k_{pr,B} \in \{1,2,\ldots,p-1\}$
compute $A = k_{pub,A} \equiv g^a \bmod p$	$k_{pub,A} = A$ →	compute $B = k_{pub,B} \equiv g^b \bmod p$
	← $k_{pub,B} = B$	
$k_{AB} = k_{pub,B}^{k_{pr,A}} \equiv B^a \bmod p$		$k_{AB} = k_{pub,A}^{k_{pr,B}} \equiv A^b \bmod p$

- 问题：为了实现加密与解密，N个用户如何交换密码——DH协议

Diffie-Hellman协议

Diffie-Hellman Set-up

1. $p = 13$.
2. $g = 7$.
3. Publish p and g.

Diffie-Hellman Key Exchange

Alice		Bob
choose $a = 2$		choose $b = 3$
compute $10 = 7^2 \ mod \ 13$		compute $5 = 7^3 \ mod \ 13$
	$k_{pub,A} = 10$ →	
	← $k_{pub,B} = 5$	
$k_{AB} = 5^2 \ mod \ 13 = 12$		$k_{AB} = 10^3 \ mod \ 13 = 12$

- 问题：为了实现加密与解密，N个用户如何交换密码——DH协议

Diffie-Hellman协议面临的窃听攻击

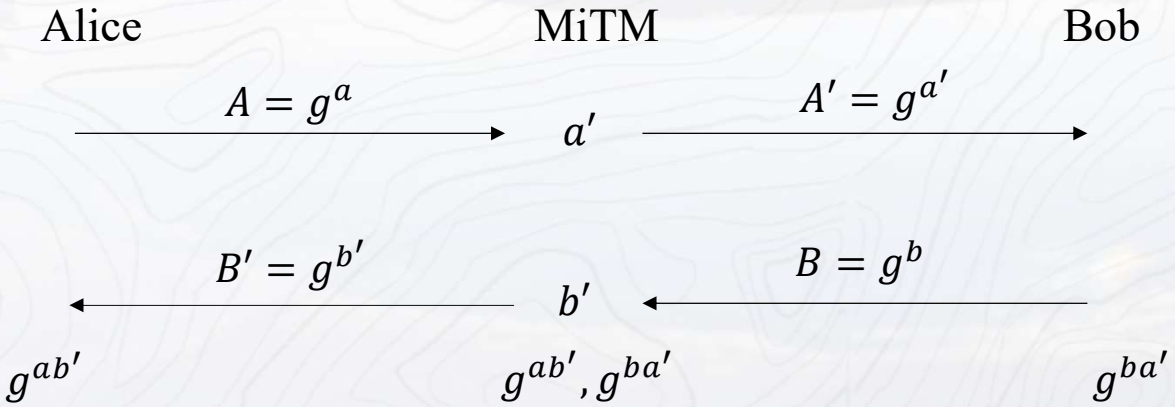
- Eavesdropper sees: p, g, A $= g^a \ mod \ p$, and B $= g^b \ mod \ p$. Can she compute $g^{ab} mod \ p$?
- More generally: define $DH_g(\alpha^a, \alpha^b) = \ g^{ab} \ mod \ p$. How hard is the DH function mod p?

普通数域筛法：指数级时间$\exp(\tilde{O}(\sqrt[3]{n}))$
假设质数足够大

cipher key size	modulus size	Elliptic Curve size
80 bits	1024 bits	160 bits
128 bits	3072 bits	256 bits
256 bits (AES)	15360 bits	512 bits

- 问题：为了实现加密与解密，N个用户如何交换密码——**DH协议**

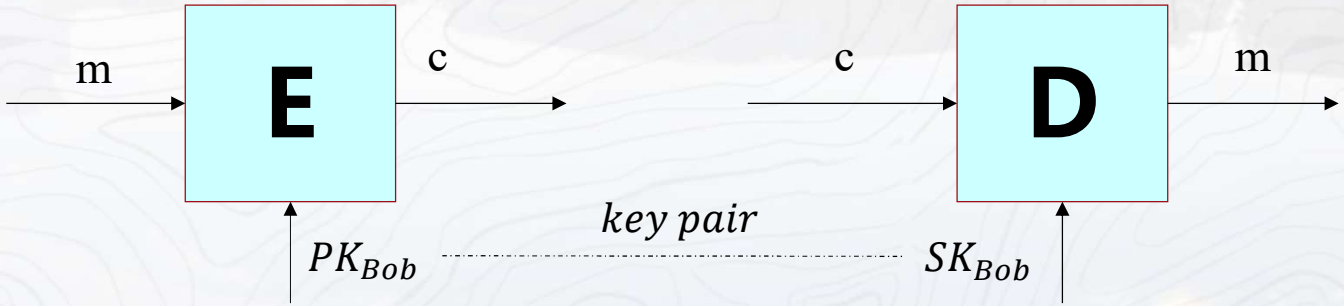
Diffie-Hellman协议面临的中间人攻击



Diffie-Hellman协议的非互动性

微博			
g^a	g^b	g^c	g^d
Alice	Bob	Charlie	David
a	b	c	d
$k_{ac} = g^{ac}$		$k_{ac} = g^{ac}$	

- 问题：为了实现加密与解密，N个用户如何交换密码——公钥加密系统



$c = E(PK_{Bob}, m)$

$PK: public\ key$
$SK: secret\ key$

定义: 一个公钥加密系统是算法的三元组 (G, E, D)

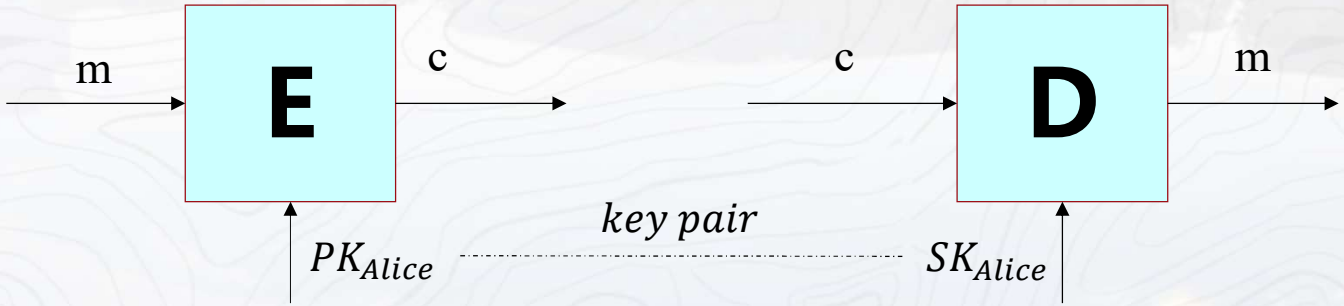
$G()$: 随机算法，输出一对密钥$(pk,\ sk)$ ——密钥生成算法

$E(pk, m)$: 随机算法，输入$m \in M$ 输出 $c \in C$ ——加密算法

$D(sk, c)$: 输入$c \in C$输出$m \in$ M or $\perp$ ——解密算法

一致性: 任意由$G()$生成的密钥对都满足任意$m \in M$: $D(sk, E(pk, m)) = m$，即 使用公钥加密明文然后用私钥解密可以获得最初的明文。

- 问题：为了实现加密与解密，N个用户如何交换密码——公钥加密系统



$c = E(PK_{Alice}, m)$

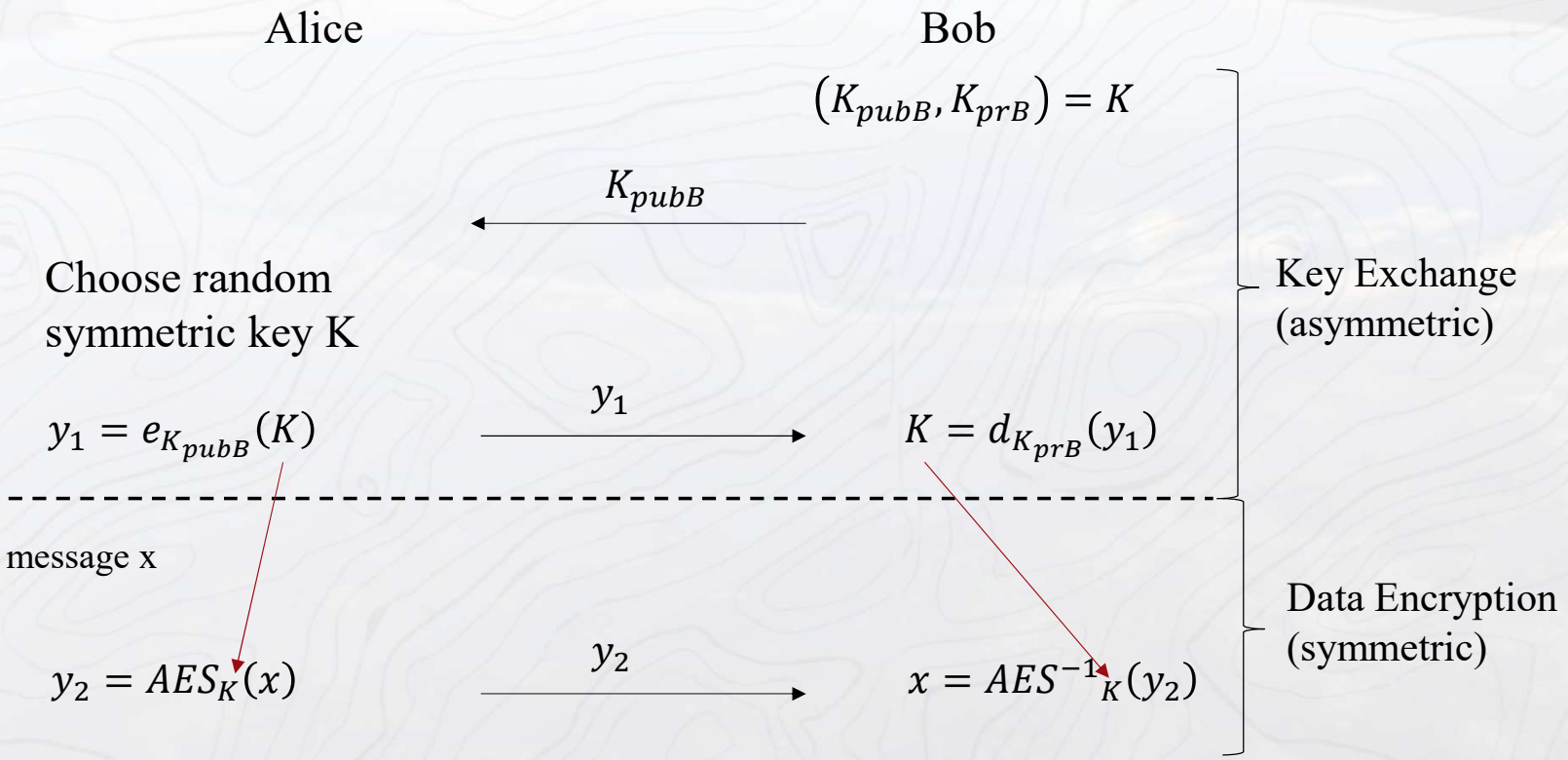
$PK: public\ key$
$SK: secret\ key$

1. Alice生成随机的公钥、私钥，并将公钥发送给Bob
2. Bob用Alice的公钥对信息m加密得到密文c并发送给Alice
3. Alice用私钥对c解密得到信息m，由此m可以作为双方后续通信中的密钥

与DH协议的不同：双方必须有先后的通信，即收到Alice发送的公钥后，Bob才能开始加密传输消息

- 问题：为了实现加密与解密，N个用户如何交换密码——公钥加密系统

Example: Hybrid protocol with AES as the symmetric cipher




Any Questions?

RSA与ECC

- **RSA：1977年 Rivest、Shamir and Adleman**

In a "public-key cryptosystem" each user places in a public file an encryption procedure E. That is, the public file is a directory giving the encryption procedure of each user. The user keeps secret the details of his corresponding decryption procedure D. These procedures have the following four properties:

(a) Deciphering the enciphered form of a message M yields M. Formally,

$$D(E(M)) = M. \quad (1)$$

(b) Both E and D are easy to compute.

(c) By publicly revealing E the user does not reveal an easy way to compute D. This means that in practice only he can decrypt messages encrypted with E, or compute D efficiently.

(d) If a message M is first deciphered and then enciphered, M is the result. Formally,

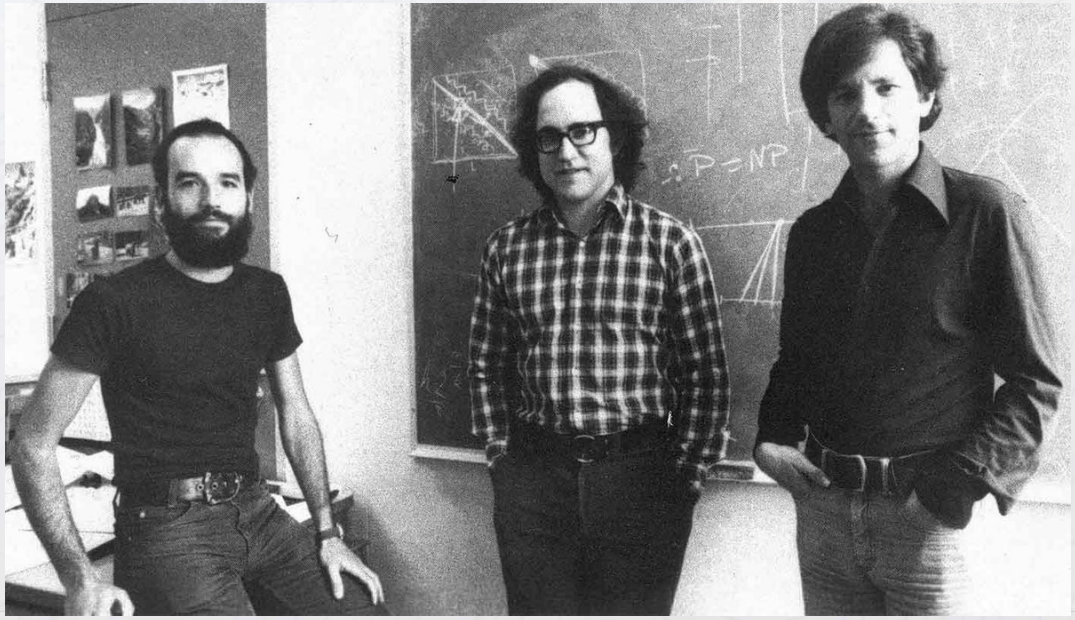
$$E(D(M)) = M. \quad (2)$$

Programming Techniques — S.L. Graham, R.L. Rivest* Editors

A Method for Obtaining Digital Signatures and Public-Key Cryptosystems

R. L. Rivest, A. Shamir, and L. Adleman
MIT Laboratory for Computer Science and Department of Mathematics

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, *21*(2), 120-126.



- **RSA**：密钥生成

算法基于一个简单的数论知识：给出两个素数，很容易将它们相乘，然而给出它们的乘积，想得到这两个素数就显得尤为困难。 $N = p \times q$?

Algorithm: RSA Key Generation

公钥: $kpub = (n, e)$ and 私钥$kpr = d$

1. 选择两个大质数 p, q
2. 计算 $n = p * q$
3. 计算 $\Phi(n) = (p-1) * (q-1)$
4. 选择随机数 $e \in \{1, 2, \ldots, \Phi(n) - 1\}$ 满足 $\gcd(e, \Phi(n)) = 1$
5. 计算私钥 d 满足 $d * e \equiv 1 \, mod \, \Phi(n)$
6. **RETURN** $kpub = (n, e), kpr = d$

- **RSA**：密钥生成

例：

1. 选择质数61和53
2. 计算$n = 61 * 53 = 3233$
3. 计算 $\Phi(n) = 60 * 52 = 3120$
4. 选择与3120互质且小于3120的随机数$e = 17$
5. 计算17关于3120的模反元素$d = 2753$
6. 公钥就是 (3233,17)，私钥就是(3233, 2753)

n的长度就是密钥长度。3233写成二进制是110010100001，一共有12位，所以这个密钥就是12位。实际应用中，RSA密钥一般是1024位，重要场合则为2048位。

- **RSA**：加密解密过程

对于明文x，用公钥(n, e)对x加密的过程，就是将x转换成数字（字符串的话取其 ASCII码或者 unicode 值），然后通过幂取模计算出y，其中y是密文；

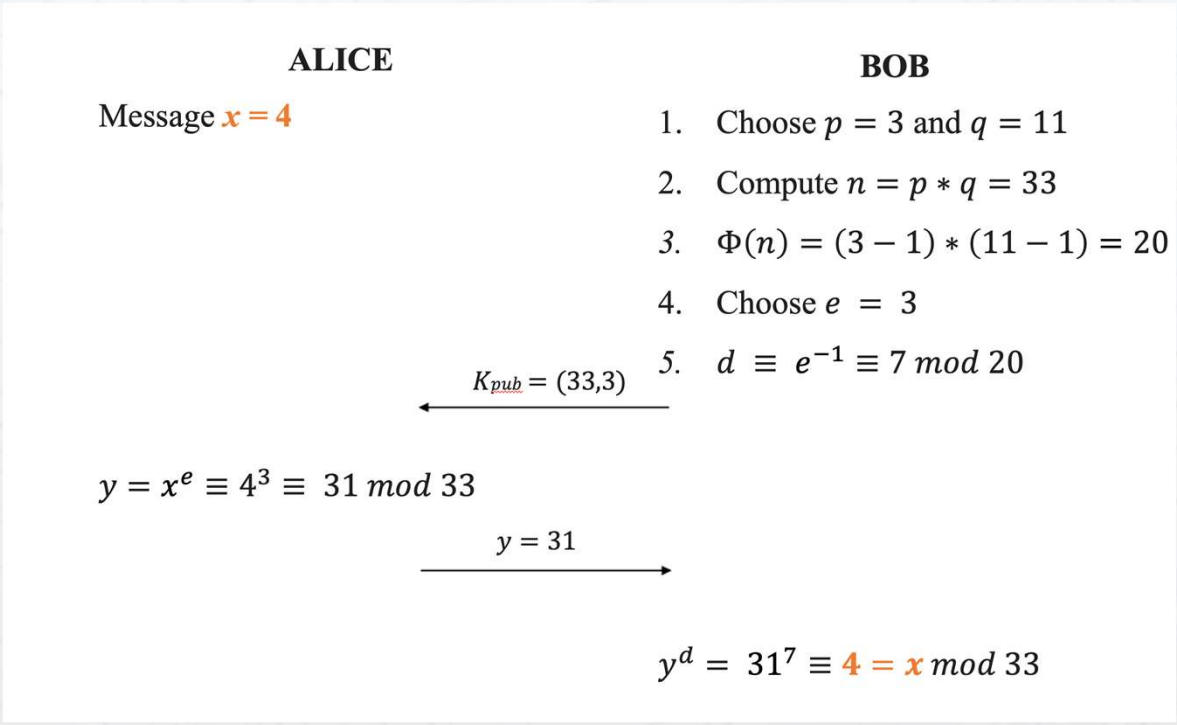
$$y = e_{k_{pub}}(x) \equiv x^e \bmod n$$

对于密文，用私钥(n, d)对y进行解密的过程和加密类似，同样是计算幂取模：

$$x = d_{k_{pr}}(y) \equiv y^d \bmod n$$

- **RSA**：密钥生成

练习：Alice 想要向 Bob 发送加密信息。Bob首先计算步骤 1-5 中的 RSA 参数。然后，他向 Alice 发送自己的公开密钥。Alice 对信息进行加密 ($x = 4$) 并将密文 y 发送给 Bob，最后Bob使用自己的私人密钥解密 y。



- **RSA**：密钥生成

练习：Alice 想要向 Bob 发送加密信息。Bob选择的参数是p=3,q=11。然后，他向 Alice 发送自己的公开密钥。Alice 对信息进行加密 ($x = \text{key}$) 并将密文 y 发送给 Bob，最后Bob使用自己的私人密钥解密 y。

要发送的明文是key，数字化：11，05，25

用(33,3)加密信息：

$$c_1 = 11^3 mod\ 33 = 11$$
$$c_2 = 05^3 mod\ 33 = 26$$
$$c_1 = 25^3 mod\ 33 = 16$$

转成字母：kzp

用(33,7)解密信息：

$$m_1 = 11^7 mod\ 33 = 11$$
$$m_2 = 26^3 mod\ 33 = 05$$
$$m_3 = 16^3 mod\ 33 = 25$$

- **RSA**：安全性证明

RSA过程中涉及的参数：$p, q, n, \phi(n), e, d$

窃听者能够获取的只有公钥(n, e)，是否能通过这些计算出明文？

1. 根据数学证明，$x = y^d \ mod \ n$，想要根据密文计算明文，必须知道私钥d；
2. d为e对$\phi(n)$的模逆元，如果能知道$\phi(n)$，则可以进一步计算出d；
3. $\phi(n) = (p-1)(q-1)$，其中p和q均为质数，只有得到p和q的值才能计算$\phi(n)$；
4. $n = pq$，n是公开的，因此如果能够分解n，则p和q就能被计算出来。

破解 **RSA** 的难点在于对n的因数分解

- **RSA：Weakness**

1. RSA 加密是确定性的，即对于特定密钥，特定明文总是映射到特定密文。
2. 明文值 $x = 0$、$x = 1$ 或 $x = -1$ 在密文中不会改变。
3. 小的随机数e 和小的明文 x 可能会受到攻击。
4. RSA 具有延展性。攻击者可以将密文转换成另一种密文，从而导致明文的已知转换。这种变化不会被接收方察觉。

延展性(Malleability)，是指给定未知消息m的密文c，可以得到未知消息m_1的密文c_1，其中m和m_1具有某种已知的关联，从而导致选择密文攻击。
例如：攻击者观察到使用公钥(N,e)加密的密文$c = m^e (mod\ N)$，则有密文$c_1 = 2^e c (mod\ N)$，解密为$2m (mod\ N)$，因为$c_1^d \equiv (2^e m^e)^d \equiv 2^{ed} m^{ed} \equiv 2m (mod\ N)$

- **RSA：Weakness**

1. RSA 加密是确定性的，即对于特定密钥，特定明文总是映射到特定密文。
2. 明文值 $x = 0$、$x = 1$ 或 $x = -1$ 在密文中不会改变。
3. 小的随机数e 和小的明文 x 可能会受到攻击。
4. RSA 具有延展性。攻击者可以将密文转换成另一种密文，从而导致明文的已知转换。这种变化不会被接收方察觉。
5. 计算量大，效率低。

事实上，RSA 极高的计算要求严重阻碍了其发明后的实际应用。在 20 世纪 70 年代的计算机上，进行数十万次整数乘法是不可能的。虽然如今在高速硬件上进行一次 RSA 运算的时间可以达到 100 μs，但与当今许多网络的速度相比，使用 RSA 对大量数据进行加密的吞吐量仍然相当缓慢。因此，RSA 和其他公钥算法不用于大量数据加密。

- **ECC椭圆曲线加密算法(Elliptic curve cryptography)**

问题

RSA 等非对称方案需要在参数超过 1000 位的整数环和字段中进行指数运算，计算量太大。

动机

需要更小的字段大小，以提供同等的安全性 。

解决方案

椭圆曲线加密法使用一组点（而不是整数）来实现系数大小为 160-256 位的加密方案，从而大大减少了计算量。

- **ECC椭圆曲线加密算法(Elliptic curve cryptography)**

椭圆曲线：在有限域上的三次方程$y^2 = x^3 + ax + b$

Definition 9.1.1 Elliptic Curve

The elliptic curve *over* $\mathbb{Z}_p$, $p > 3$, *is the set of all pairs* $(x, y) \in \mathbb{Z}_p$ *which fulfill*

$$y^2 \equiv x^3 + a \cdot x + b \bmod p \qquad (9.1)$$

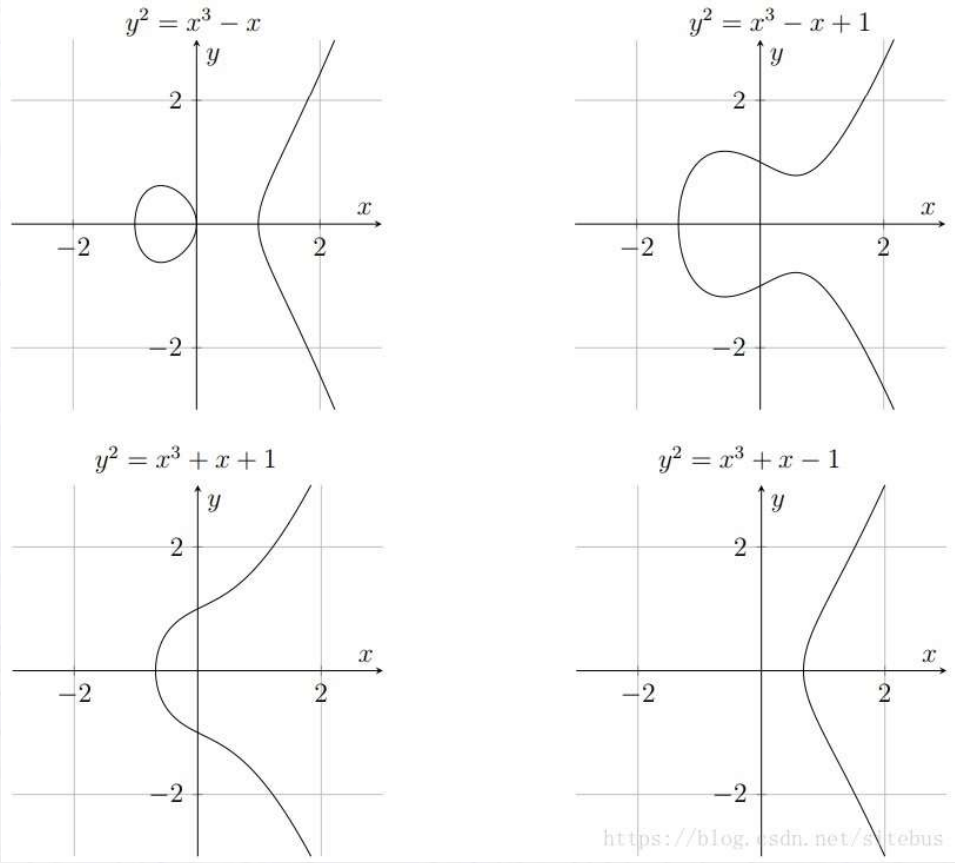
together with an imaginary point of infinity $\mathcal{O}$, *where*

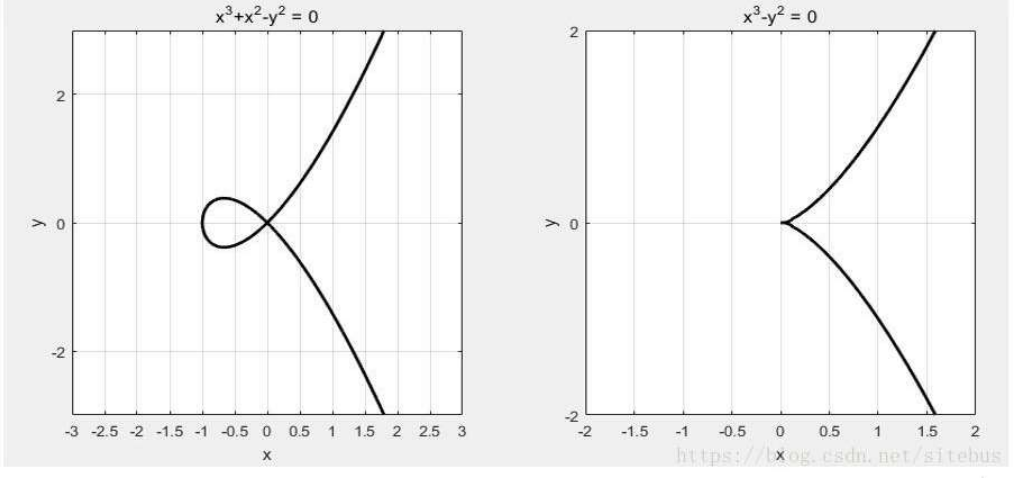
$$a, b \in \mathbb{Z}_p$$

and the condition $4 \cdot a^3 + 27 \cdot b^2 \neq 0 \bmod p$.

- **ECC椭圆曲线加密算法(Elliptic curve cryptography)**

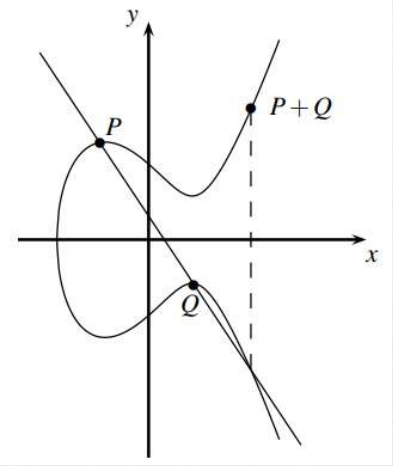
椭圆曲线：$y^2 = x^3 + ax + b$，$\Delta = -16(4a^3 + 27b) \neq 0$

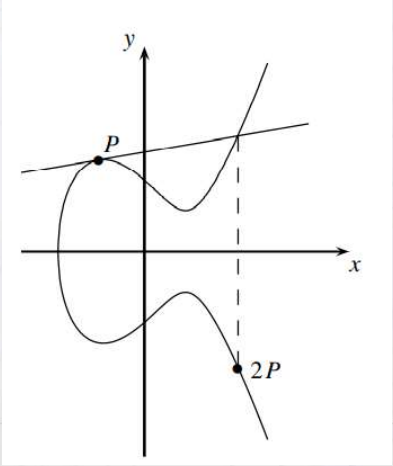


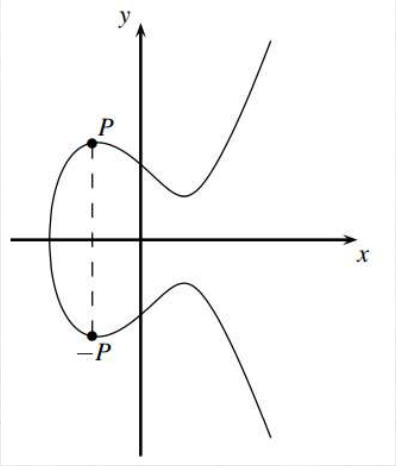


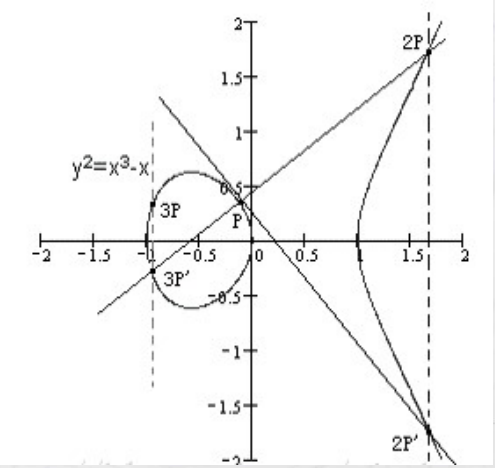
- **ECC椭圆曲线加密算法(Elliptic curve cryptography)**
  - 点加法P+Q
    1. 通过 P 和 Q 画一条直线，得到椭圆曲线与直线的第三个交点。
    2. 沿 x 轴镜像这第三个交点。
  - 同点加法P+P
    1. 画出经过 P 的切线，得到该切线与椭圆曲线的第二个交点。
    2. 沿 x 轴镜像第二个交点。
  - 逆元

    点$P=(x_p,y_p)$的逆元是点$-P=(x_p,-y_p)$，即沿 x 轴反射的点。



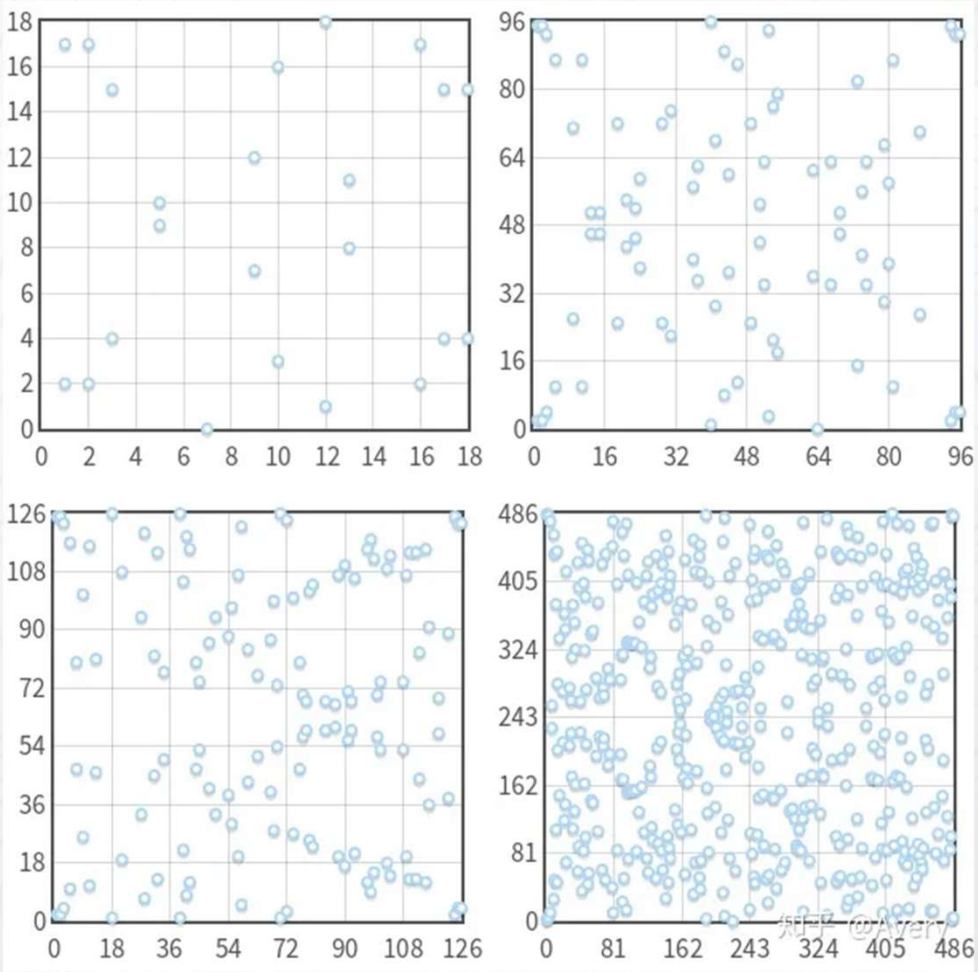






- **ECC椭圆曲线加密算法(Elliptic curve cryptography)**
  - 有限域：整数模p的集合记作F_p，p是质数。
  - 性质：加法和乘法具有封闭性，即在有限域中，两个数的加和乘的结果仍然在这个有限域中；满足交换律和结合律。
  - 以F_{23}为例：
    加：$(18+9) mod\ 23 = 4$
    减：$(7-14) mod\ 23 = 16$
    乘：$4 * 7\ mod\ 23 = 5$
    加法逆元：$-5\ mod\ 23 = 18$ 验证：$\big(5+(-5)\big) mod\ 23 = (5+18)\ mod\ 23 = 0$
    乘法逆元：$9^{-1}\ mod\ 23 = 18$ 验证：$(9 * 9^{-1}) mod\ 23 = (9 * 18)\ mod\ 23 = 1$
  - 有限域F_p上的椭圆曲线：点集$\left\{(x,y) \in \left(F_p\right)^2 \middle| y^2 \equiv x^3 + ax + b (mod\ p)\right\} \cup \{0\}$，其中$4a^3 + 27b \not\equiv 0 (mod\ p)$

- ECC椭圆曲线加密算法(Elliptic curve cryptography)



$$y^2 \equiv x^3 - 7x + 10 (mod\ p),$$
$$p = \{19,97,127,487\}$$

- **ECC椭圆曲线加密算法(Elliptic curve cryptography)**

椭圆曲线的离散对数问题

给定椭圆曲线E上的两个点$P, Q \in E$，寻找这样的一个整数a，使得$Q = aP$。

ECC算法中，a是私钥，$Q(x, y)$是公钥

Double-and-Add Algorithm for Point Multiplication

Input: elliptic curve E together with an elliptic curve point P a scalar $d = \sum_{i=0}^{t} d_i 2^i$ with $d_i \in 0,1$ and $d_t = 1$

Output: $T = dP$

Initialization: $T = P$

Algorithm:

1 FOR $i = t - 1$ DOWNTO 0

1.1 $T = T + T \bmod n$ IF $d_i = 1$

1.2 $T = T + P \bmod n$

2 RETURN (T)

- **ECC椭圆曲线加密算法(Elliptic curve cryptography)**

1. 选定一条椭圆曲线$E_p(a,b)$，并取椭圆曲线上一点，作为基点P。
2. 选择一个大数k作为私钥，并生成公钥$Q = kP$。
3. 将$E_p(a,b)$和点Q,P传给用户。
4. 用户接到信息后，将待传输的明文编码到$E_p(a,b)$上的一点M，并产生一个随机整数r。
5. 公钥加密（密文C是一个点对）：$C = \{rP, M + rQ\}$
6. 私钥解密：$M + rQ - k(rP) = M + r(kP) - k(rP) = M$
7. 对点M进行解码就可以得到明文。

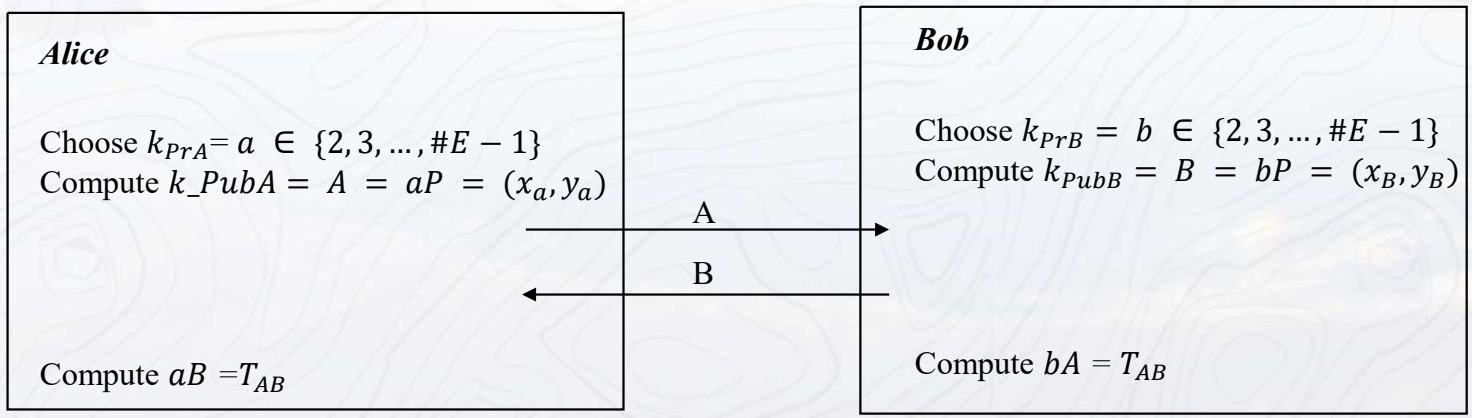
假设在加密过程中，有一个攻击者H，H只能知道椭圆曲线$E_p(a,b)$、公钥Q、基点P和密文点C，需要保密的是私钥k和随机数r。通过公钥、基点求私钥或者通过密文点、基点求随机数都是非常困难的，由此保证数据传输的安全。

- **ECC椭圆曲线加密算法(Elliptic curve cryptography)**

明文编码方式：

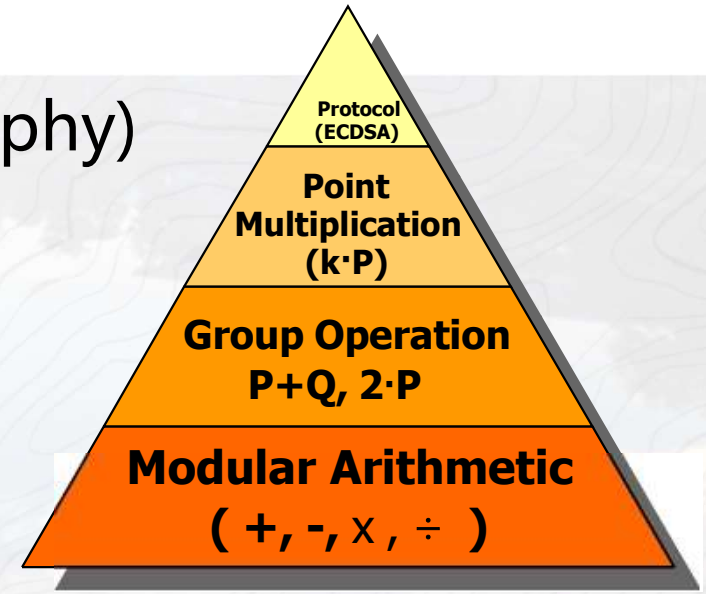
1. 将消息m转化为整数（如使用ASCII码表，可以将HELLO转化为7269767679）。
2. 尝试为该数寻址一个在椭圆曲线上的点，横坐标x从这个数字开始，逐步增加并计算椭圆曲线公式对应的y，直到找到一个有效的y（即能开平方）。

- **ECC椭圆曲线加密算法(Elliptic curve cryptography)**
  - 密钥交换：与DH协议相同



  - Joint secret between Alice and Bob: $T_{AB} = (x_{AB}, y_{AB})$.
  - Since point addition is associative, the correctness of the protocol is easy to prove.

- **ECC椭圆曲线加密算法(Elliptic curve cryptography)**



➢ 四层结构：

1. 在底层进行模块运算，即素数域 GF(p) 的运算。
2. 在下一层，实现两个分组运算，即点加倍和点加法。
3. 第三层实现标量乘法，使用上一层的分组运算。
4. 顶层实现实际协议，如 ECDH 或 ECDSA。

➢ 软件实现：

在 3GHz 64 位 CPU 上优化的 256 位 ECC 实现，每次点乘法需要约 2 毫秒
功能较弱的微处理器（如智能卡或手机）甚至需要更长的时间（>10 毫秒）

➢ 硬件实现：

使用 256 位特殊素数的高性能实现可在可重新配置的硬件上在几百微秒内计算一个点乘法
运算用于 ECC 的专用芯片甚至可以在几十微秒内计算一个点乘法运算

- ECC椭圆曲线加密算法(Elliptic curve cryptography)

RSA	ECC
一种成熟的公用密钥加密方法。	与 RSA 相比是一种较新的公钥加密方法。
基于质因数分解法的原理。	基于椭圆曲线的数学表示。
RSA 运行速度比 ECC 快，这得益于它的简单性。	ECC 需要更多时间，因为它本质上很复杂。
RSA 已被发现存在漏洞，并即将寿终正寝。	ECC 比 RSA 更安全，目前正处于适应阶段。预计在不久的将来，它的使用范围将会扩大。
RSA 需要更大的密钥长度来实现加密。	与 RSA 相比，ECC 所需的密钥长度要短得多。

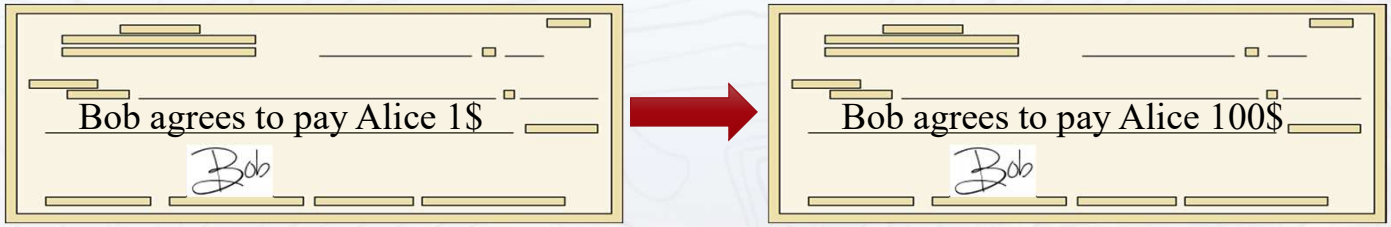

Any Questions?

数字签名与MAC

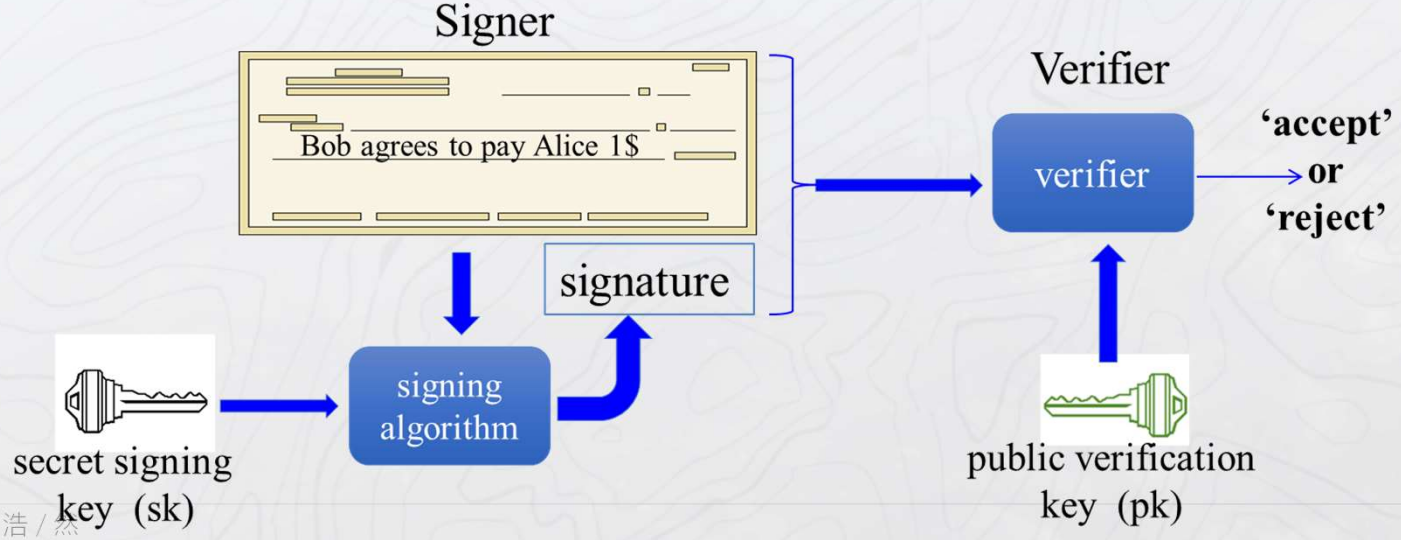
- 数字签名

目标：实现从文件到作者的绑定

问题：文档中的签名很容易被复制到其他文档中

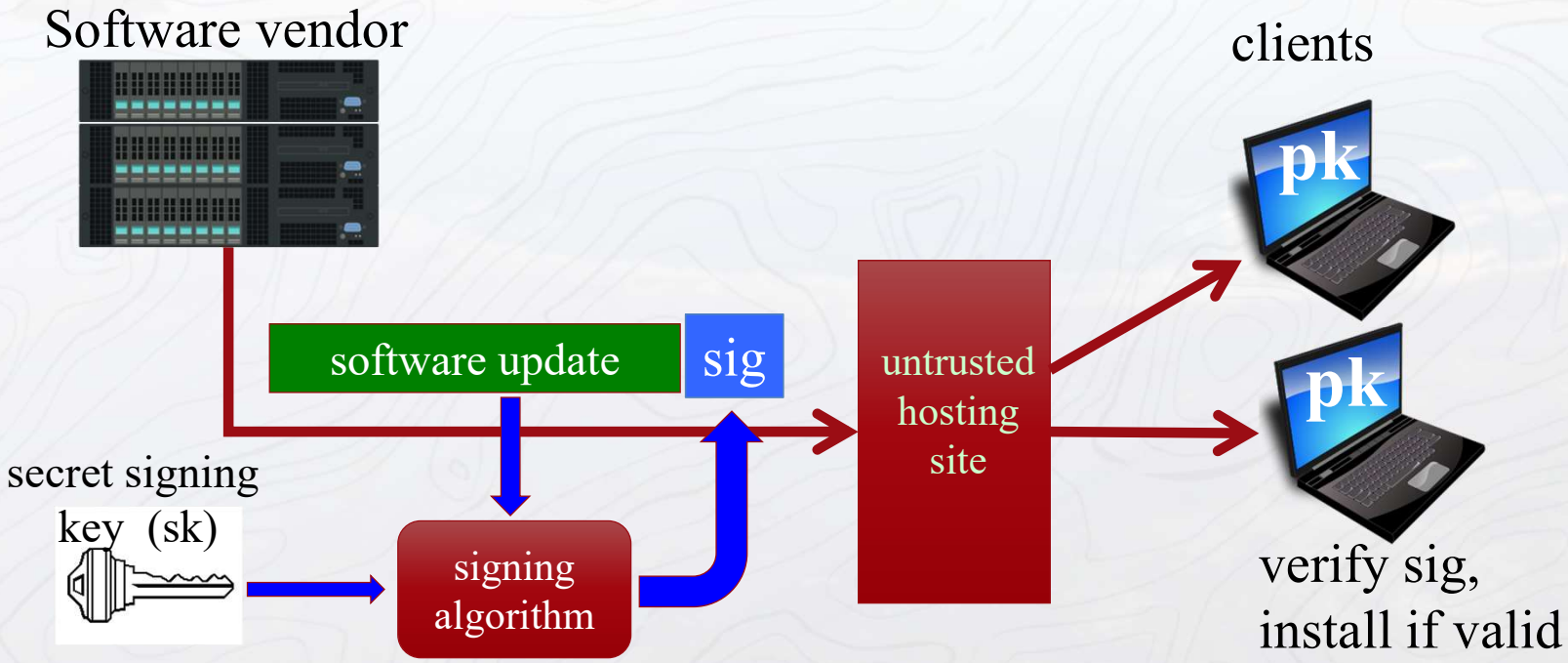


解决方案：让签名与文档内容相关联



- 数字签名

例：软件分发


Software vendor
clients
pk
pk
software update
sig
untrusted
hosting
site
secret signing
key (sk)
signing
algorithm
verify sig,
install if valid

- 数字签名

签名协议

Basic Digital Signature Protocol

Alice		Bob
		genrate $k_{pr,B}, k_{pub,B}$
	$k_{pub,B}$ ←	publish public key
		sign message
		$s = sig_{k_{pr,B}}(x)$
	(x, s) ←	send message + signature

verify signature:

$ver_{k_{pu\ ,B}}(x, s) = true/false$

- 数字签名

基于RSA的签名协议

RSA Keys
Bob's private key: $k_{pr} = (d)$
Bob's public key: $k_{pub} = (n,\ e)$

Basic RSA Digital Signature Protocol

Alice　　Bob

$k_{pr} = d, k_{pub} = (n, e)$

(n, e) ←

compute signature:
$s = sig_{k_{pr}}(x) \equiv x^d \bmod n$

(x, s) ←

verify: $ver_{k_{pub}}(x, s)$
$x' \equiv s^e \bmod n$

x' { $\equiv x \bmod n$ => valid signature
$\not\equiv x \bmod n$ => invalid signature

- 数字签名

基于RSA的签名协议——示例

Alice

Bob

1. $choose\ p = 3\ and\ q = 11$
2. $n = ?$
3. $\phi(n) = ?$
4. $choose\ e = 3$
5. $d \equiv e^{-1} \equiv ?\ mod\ 20$

$(n, e) = (?, 3)$ ←

compute signature for message:
$x = 4$
$s = ?$

$(x, s) = (4, ?)$ ←

verify:
$x' = s^e \equiv ?$
$x' \equiv x\ mod\ 33$ => valid signature

- 数字签名

基于RSA的签名协议——示例

Alice

Bob

1. $choose\ p = 3\ and\ q = 11$
2. $n = p \cdot q = 33$
3. $\phi(n) = (3-1)(11-1) = 20$
4. $choose\ e = 3$
5. $d \equiv e^{-1} \equiv 7\ mod\ 20$

$(n, e) = (33,3)$ ←

compute signature for message:
$x = 4$
$s = x^d \equiv 4^7 \equiv 16\ mod\ 33$

$(x, s) = (4,16)$ ←

verify:
$x' = s^e \equiv 16^3 \equiv 4\ mod\ 33$
$x' \equiv x\ mod\ 33$ => valid signature

- 数字签名

伪造攻击：

Existential Forgery

Alice	**Oscar**	**Bob**
		$k_{pr} = d$ $k_{pub} = (n, e)$
$\leftarrow (n, e)$		$\leftarrow (n, e)$
	1.choose signature: $s \in \mathbb{Z}_n$ 2.compute message: $x \equiv s^e \ mod \ n$	
$\leftarrow (x, s)$		

verification:
$s^e \equiv x' mod\ n$
since $x' = x$
=> valid signature!

攻击者首先选择签名，然后计算信息。因为不知道私钥，他无法控制信息x的语义。例如， **Oscar**无法生成 "向**Oscar**账户转账 1000 美元 "这样的信息。

- 数字签名——基于离散对数理论

椭圆曲线的离散对数问题

给定椭圆曲线E上的两个点$P, Q \in E$，寻找这样的一个整数a，使得$Q = aP$。

离散对数问题

欧拉函数：$\varphi(n)$是小于或者等于n的正整数中与n互质的数的数目。

例：$n = 7, \varphi(7) = 6; n = 10, \varphi(10) = 4$

欧拉定理：若n, a为正整数，且n, a互质，则$a^{\varphi(n)} \equiv 1(mod\ n)$。

简单来说就是：a的$\varphi(n)$次方除以n得到的余数等于1

例：

$$a = 3, n = 7: 3^{\varphi(7)} = 3^6 = 729, \quad 729 \div 7 = 104 \cdots\cdots 1$$

$$a = 6, n = 10: 6^{\varphi(10)} = 6^4 = 1296, \quad 1296 \div 7 = 185 \cdots\cdots 1$$

- 数字签名——基于离散对数理论

椭圆曲线的离散对数问题

给定椭圆曲线E上的两个点$P, Q \in E$，寻找这样的一个整数a，使得$Q = aP$。

离散对数问题

欧拉函数：$\varphi(n)$是小于或者等于n的正整数中与n互质的数的数目。

欧拉定理：若n, a为正整数，且n, a互质，则$a^{\varphi(n)} \equiv 1(mod\ n)$。

若n, a为正整数，且n, a互质，令$a^d \equiv 1(mod\ n)$，如果用$\delta(n, a)$表示使该式成立的最小的正整数d，此时如果$\delta(n, a) = \varphi(n)$，则称$a$为模$n$的原根。

例：

若$a = 3, n = 7, \varphi(7) = 6, 3^6 \equiv 1\ mod\ 7$，可以计算出$3^1, 3^2, 3^3, 3^4, 3^5\ mod\ 7$都不为1，$\delta(7,3) = \varphi(7)$，即$a = 3$是$n = 7$的原根。

若$a = 2, n = 7, \varphi(7) = 6, 2^3 \equiv 1\ mod\ 7$，虽然$2^6 \equiv 1\ mod\ 7$也成立，但$\delta(7,2) = 3 \neq \varphi(7) = 6$，即$a = 2$不是$n = 7$的原根。

- 数字签名——基于离散对数理论

椭圆曲线的离散对数问题

给定椭圆曲线E上的两个点$P, Q \in E$，寻找这样的一个整数a，使得$Q = aP$。

离散对数问题

欧拉函数：$\varphi(n)$是小于或者等于n的正整数中与n互质的数的数目。

欧拉定理：若n, a为正整数，且n, a互质，则$a^{\varphi(n)} \equiv 1(mod\ n)$。

若n, a为正整数，且n, a互质，令$a^d \equiv 1(mod\ n)$，如果用$\delta(n, a)$表示使该式成立的最小的正整数d，此时如果$\delta(n, a) = \varphi(n)$，则称$a$为模$n$的原根。

如果对于一个整数b和质数p的一个**原根**a，可以找到一个唯一的指数i，使得：$b = a^i(mod\ p), 0 \leq i \leq p - 1$成立，那么指数$i$称为$b$的以$a$为基数的模$p$的**离散对数**。

当已知一个大质数p和它的一个原根a，如果给定一个b，要计算i的值是相当困难的。

- 数字签名——基于离散对数理论

Key Generation for Elgamal Digital Signature

1. Choose a large prime p.
2. Choose a primitive element α of $\mathbb{Z}_P^*$ or a subgroup of $\mathbb{Z}_P^*$.
3. Choose a random integer $d \in \{2,3,\dots,p-2\}$.
4. Compute $\beta = \alpha^d \bmod p$.

The public key is now formed by $k_{pub} = (p, \alpha, \beta)$, and the private key by $k_{pr} = d$.

Elgamal Signature Generation

1. Choose a random ephemeral key $k_E \in \{0,1,2,\dots,p-2\}$ such that $\gcd(k_E, p-1) = 1$.
2. Compute the signature parameters:

$$r \equiv \alpha^{k_E} \bmod p,$$
$$s \equiv (x - d \cdot r) k_E^{-1} \bmod p - 1.$$

Elgamal Signature Verification

1. Compute the value

$$t \equiv \beta^r \cdot r^s \bmod p$$

2. The verification follows from:

$$t \begin{cases} \equiv \alpha^x \bmod p \Rightarrow valid\ signature \\ \not\equiv \alpha^x \bmod p \Rightarrow invalid\ signature \end{cases}$$

- 数字签名——基于离散对数理论

Alice

Bob

1. $choose\ p = 29$
2. $choose\ \alpha = 2$
3. $choose\ d = 12$
4. $\beta = \alpha^d = 7\ mod\ 29$

$(p, \alpha, \beta) = (29,2,7)$ ←

compute signature for message:
$x = 26$:
$choose\ k_E = 5,\ note\ that$
$\gcd(5,28) = 1$
$r = \alpha^{k_E} \equiv 2^5 \equiv 3\ mod\ 29$
$s = (x - dr)k_E^{-1} \equiv (-10) \cdot 17 \equiv 26\ mod\ 28$

$(x, (r, s)) = (26, (3,26))$ ←

verify:
$t \equiv \beta^r \cdot r^s \equiv 7^3 \cdot 3^{26} \equiv 22\ mod\ 29$
$\alpha^x \equiv 2^{26} \equiv 22\ mod\ 29$
$t \equiv \alpha^x\ mod\ 29$ => valid signature

- 数字签名——基于离散对数理论

Existential Forgery Attack Against Elgamal Digital Signature

Alice	Oscar	Bob
		$k_{pr} = d$ $k_{pub} = (p, \alpha, \beta)$
$\leftarrow (p, \alpha, \beta)$	$\leftarrow (p, \alpha, \beta)$	
	1.select integers i,j $where\ gcd(j, p-1) = 1$ 2.compute signature: $r \equiv \alpha^i \beta^j \ mod\ p$ $s \equiv -rj^{-1} \ mod\ p-1$	
$\leftarrow (x, s)$	3.compute message: $x \equiv si \ mod\ p-1$	

verification:

$t \equiv \beta^r r^s \ mod\ p$

since $t \equiv \alpha^x \ mod\ p$:

=> valid signature!

- 数字签名——基于椭圆曲线 (ECDSA)

Key Generation for ECDSA

1. Use an elliptic curve E with
- modulus p
- coefficients a and b
- a point A which generates a cyclic group of prime order q

2. Choose a random integer d with $0 < d < q$.
3. Compute $B = dA$.

The keys are now:

$k_{pub} = (p, a, b, q, A, B)$

$k_{pr} = (d)$

- 数字签名——基于椭圆曲线 (ECDSA)

ECDSA Signature Generation

1. Choose an integer as random ephemeral key k_E with $0 < k_E < q$.
2. Compute $R = k_E A$.
3. Let $r = x_R$.
4. Compute $s \equiv (h(x) + d \cdot r)k_E^{-1} \bmod q$.

ECDSA Signature Verification

1. Compute auxiliary value $w \equiv s^{-1} \; mod \; q$
2. Compute auxiliary value $u_1 \equiv w \cdot h(x) \; mod \; q$
3. Compute auxiliary value $u_2 \equiv w \cdot r \; mod \; q$
4. Compute $P = u_1 A + u_2 B$
5. The verification $ver_{k_{pub}}\big(x, (r, s)\big)$ follows from:

$$x_P \begin{cases} \equiv r \; mod \; q \Rightarrow \text{valid signature} \\ \not\equiv r \; mod \; q \Rightarrow \text{invalid signature} \end{cases}$$

- 数字签名——基于椭圆曲线 (ECDSA)

Alice

Bob

$choose\ E\ with\ p = 17, a = 2,$
$b = 2, and\ A = (5,1)\ with\ q = 19$
$choose\ d = 7$
$compute\ B = dA$
$= 7 \cdot (5,1) = (0,6)$

$(p, \alpha, \beta) = (29,2,7)$

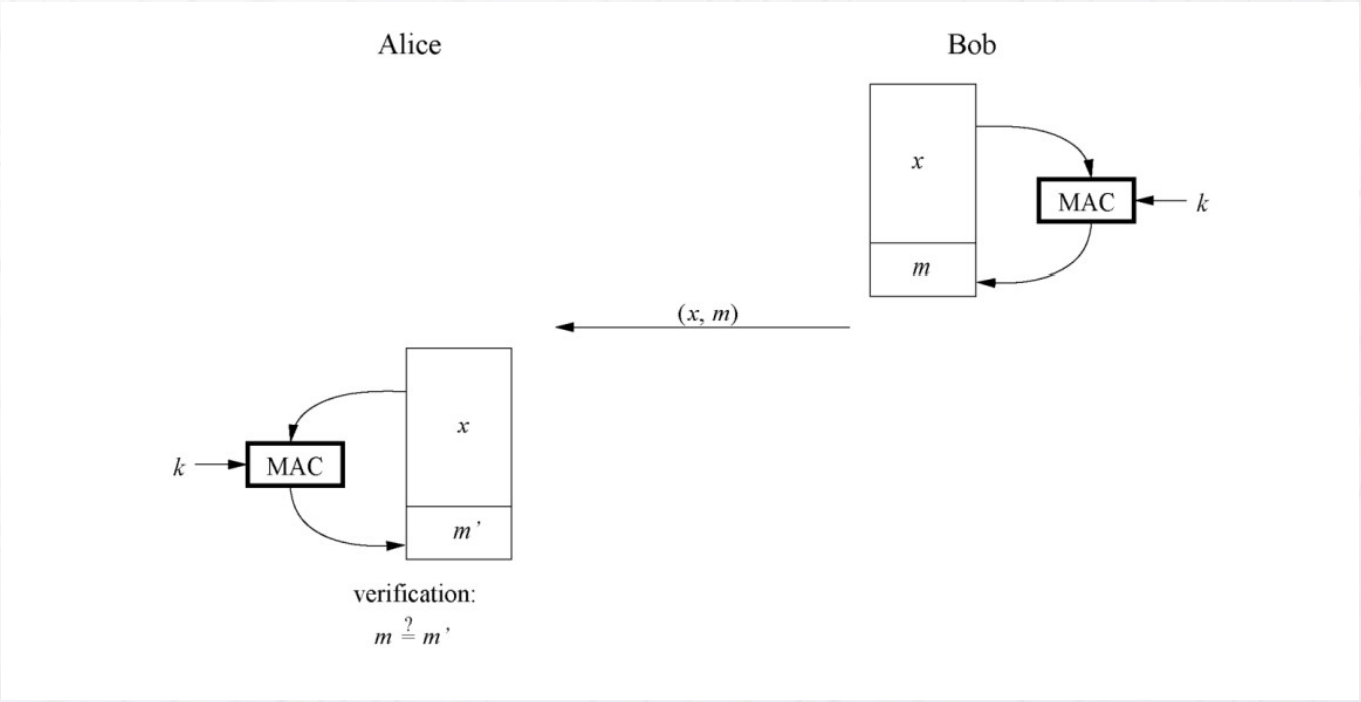
sign:
compute hash of message:
$\mathrm{h}(x) = 26$
$choose\ ephemeral\ key\ k_E = 10$
$R = 10 \cdot (5,1) = (7,11)$
$\mathrm{r} = x_R \equiv 7$
$s = (26 + 7 \cdot 7) \cdot 2 \equiv 17\ mod\ 19$

$(x, (r, s)) = (26, (3,26))$

verify:
1. $w = 17^{-1} \equiv 9\ mod\ 19$
2. $u_1 = 9 \cdot 26 \equiv 6\ mod\ 19$
3. $u_2 = 9 \cdot 7 \equiv 6\ mod\ 19$
4. $P = 6 \cdot (5,1) + 6 \cdot (0,6) = (7,11)$
5. $x_P \equiv r\ mod\ 19$ =>valid signature

$The\ elliptic\ curve\ E: y^2 \equiv x^3 + 2x + 2\ mod\ 17$

- **MAC——消息校验码Message Authentication Codes**



与数字签名类似，MAC在数据中附加了一个验证标记
MAC使用对称密钥
计算方式： $m = MAC_k(x)$

- **MAC**
  - 加密校验和： MAC 为给定信息生成加密安全的验证标记。
  - 对称： MAC 基于秘密对称密钥。签名方和验证方必须共享一个密钥。
  - 任意信息大小： MAC 接受任意长度的信息。
  - 固定输出长度： MAC 生成固定大小的验证标记。
  - 信息完整性： MAC 提供信息完整性： 在传输过程中对信息的任何篡改都会被接收者发现。
  - 信息认证： 接收方确信信息的来源。
  - 无不可抵赖性： 由于 MAC 基于对称原则，因此不提供不可抵赖性。

- **MAC**

组成MAC的三个算法：

密钥生成算法：输入安全参数，输出密钥

标记生成算法：输入密钥和消息，输出附加的标记

确定性验证算法：输入密钥、消息和标记，输出确定性的结果

生成标记的三类方法：

分组密码

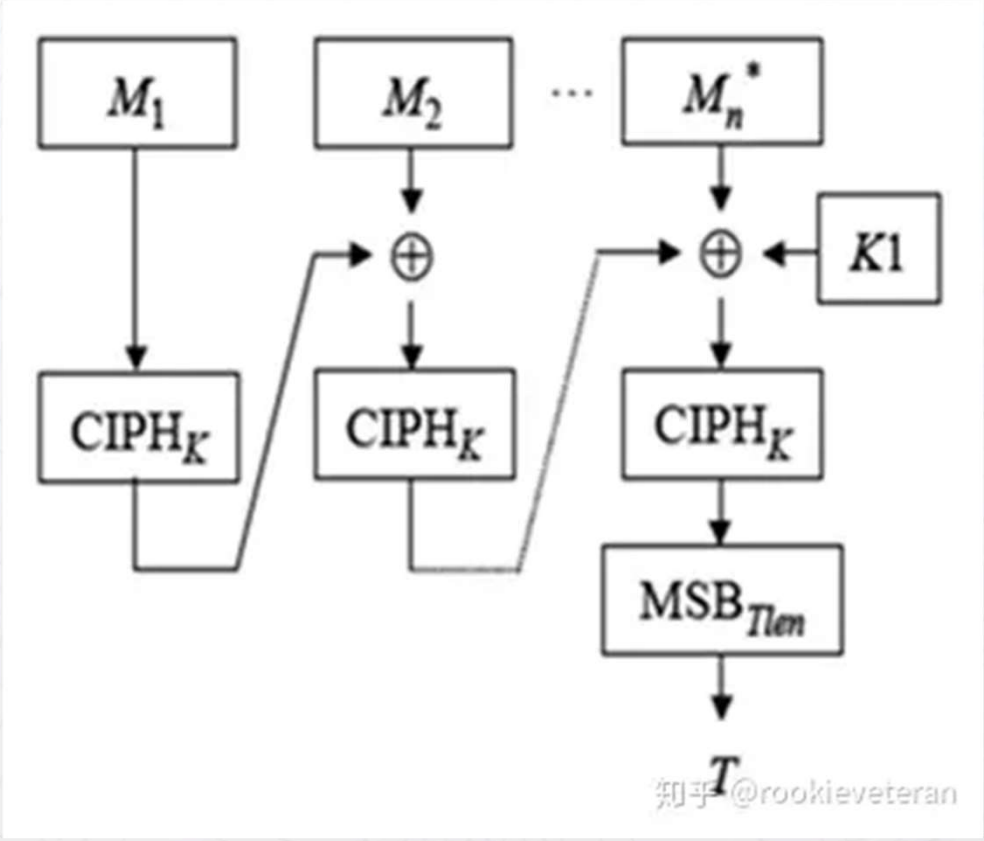
带密钥的Hash函数

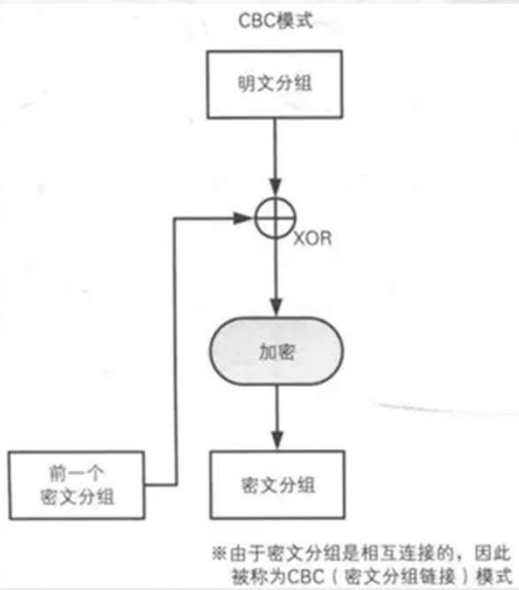
泛Hash函数

- **MAC——分组密码**

对消息使用CBC模式进行加密，取密文的最后一块作为认证码（相当于用异或操作进行了压缩）

CBC模式(Cipher Block Chaining)：首先将明文分组与前一个密文分组进行XOR运算，然后再进行加密。

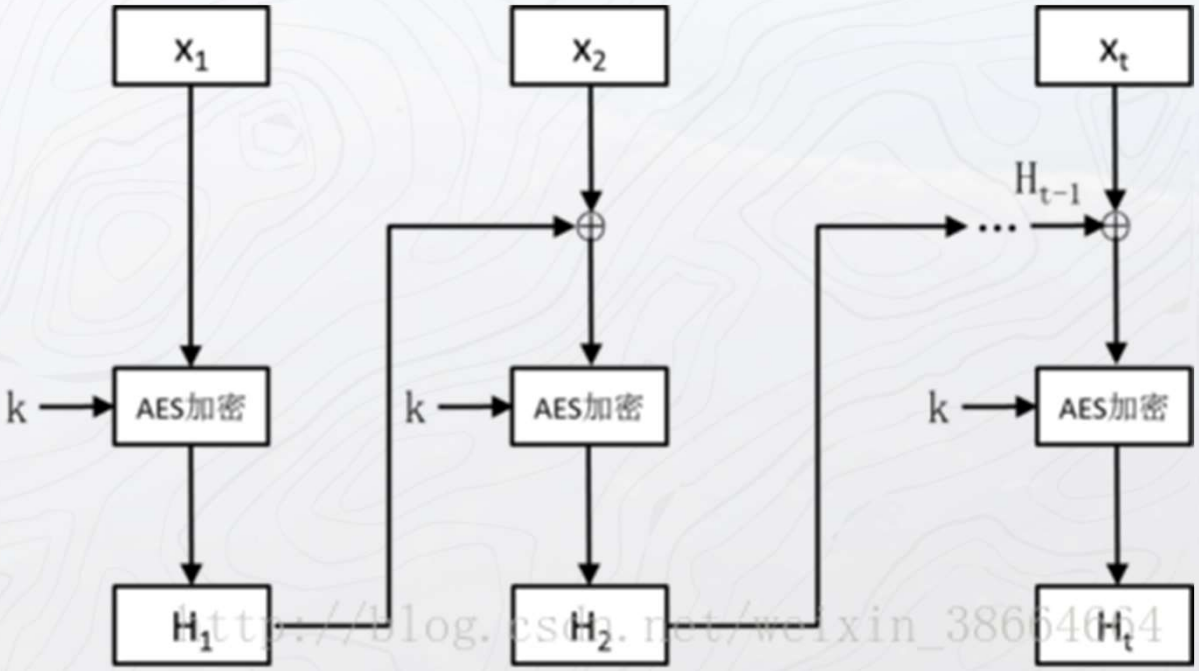


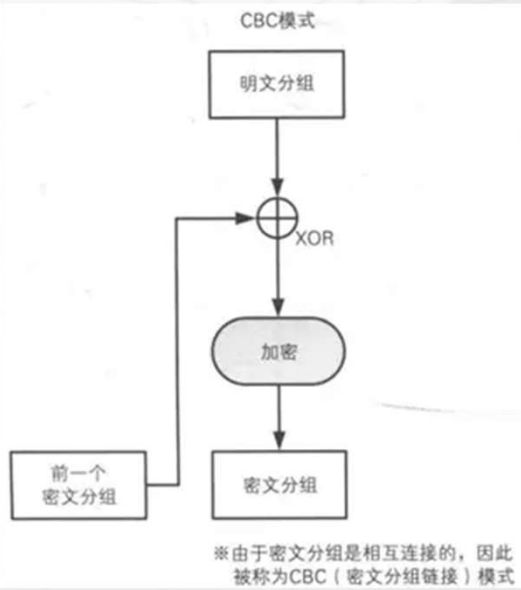


- **MAC——分组密码**

对消息使用CBC模式进行加密，取密文的最后一块作为认证码（相当于用异或操作进行了压缩）

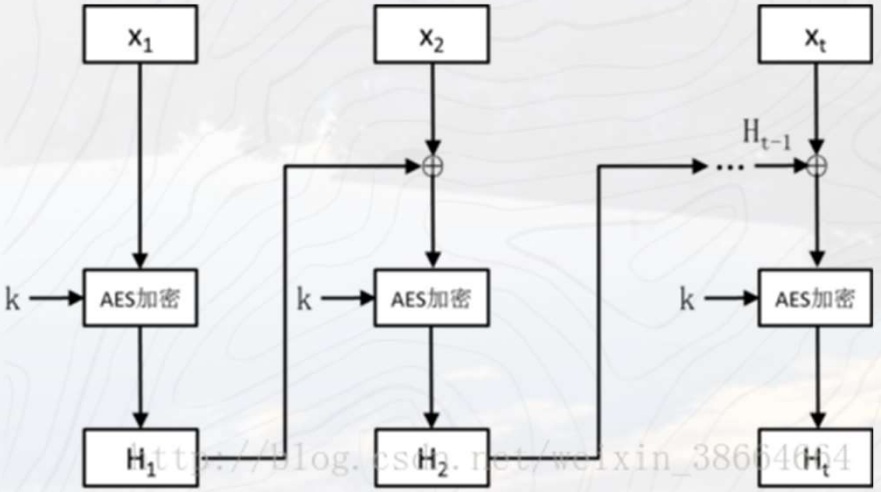
CBC模式(Cipher Block Chaining)：首先将明文分组与前一个密文分组进行XOR运算，然后再进行加密。





- **MAC——分组密码**

对消息使用CBC模式进行加密，取密文的最后一块作为认证码（相当于用异或操作进行了压缩）



例：One is always on a strange road, watching strange scenery and listening to strange music. Then one day, you will find that the things you try hard to forget are already gone.

1. 4f6e6520697320616c77617973206f6e
2. 206120737472616e676520726f61642c
3. 207761746368696e6720737472616e67
4. 65207363656e65727920616e64206c69
5. 7374656e696e6720746f20737472616e
6. 6765206d757369632e205468656e206f
7. 6e65206461792c20796f752077696c6c
8. 2066696e6420746861742074686520746
9. 68696e677320796f7520747279206861
10. 726420746f20666f726765742061726 5
11. 20616c726561647920676f6e652e**0000**

第一组x1：4f6e6520697320616c77617973206f6e
AES加密H1：3B57D91FC3273F759A360090D6B5DA93

第二组x2：206120737472616e676520726f61642c
和H1异或：1b36f96cb7555e1bfd5320e2b9d4bebf
AES加密H2：087948ED374AFEABF586DF1582658700

以此类推，得到最终的Ht

- 数字签名

什么时候使用数字签名和MAC?

- 如果一方签名，一方验证：使用 MAC

  通常需要交互来生成共享密钥

  接收方可以修改数据并重新签名，然后再将数据传递给第三方

- 如果一方签名，多方验证：使用签名

  收件人在将数据传递给第三方之前不能修改收到的数据（不可抵赖性）

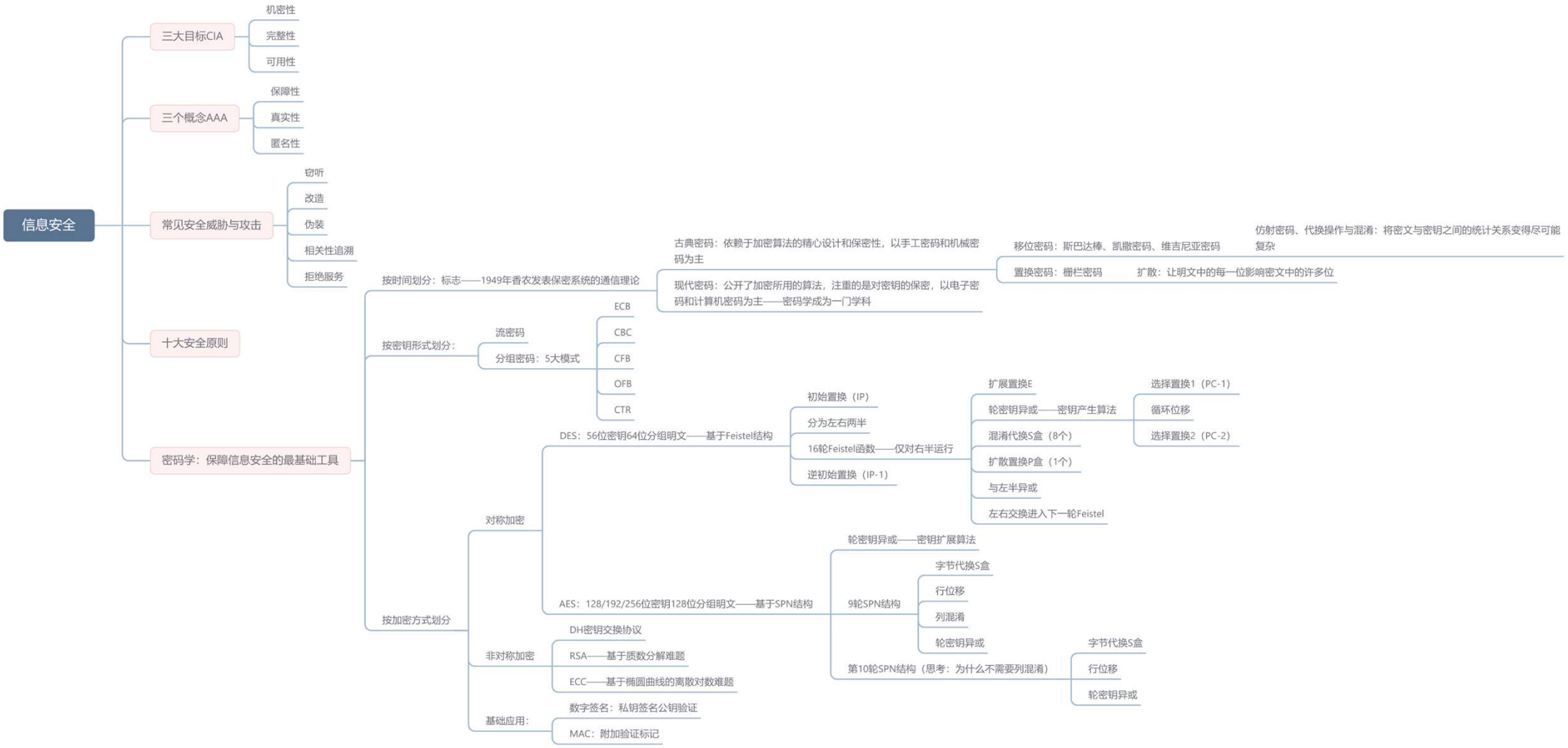
- 三种保障数据完整性的方案
  - Hash：需要抗碰撞的哈希函数——只读的公共空间
  - 数字签名：供应商必须管理长期秘钥
    供应商在软件上的签名与软件一起打包
    软件可从不受信任的分发网站下载
  - MAC：供应商必须为每个客户计算新的软件 MAC
    必须管理长期秘钥（以生成每个客户的 MAC 密钥）

	Hash	MAC	Digital signature
Integrity	Yes	Yes	Yes
Authentication	No	Yes	Yes
Non-repudiation	No	No	Yes
Kind of keys	None	Symmetric	Asymmetric

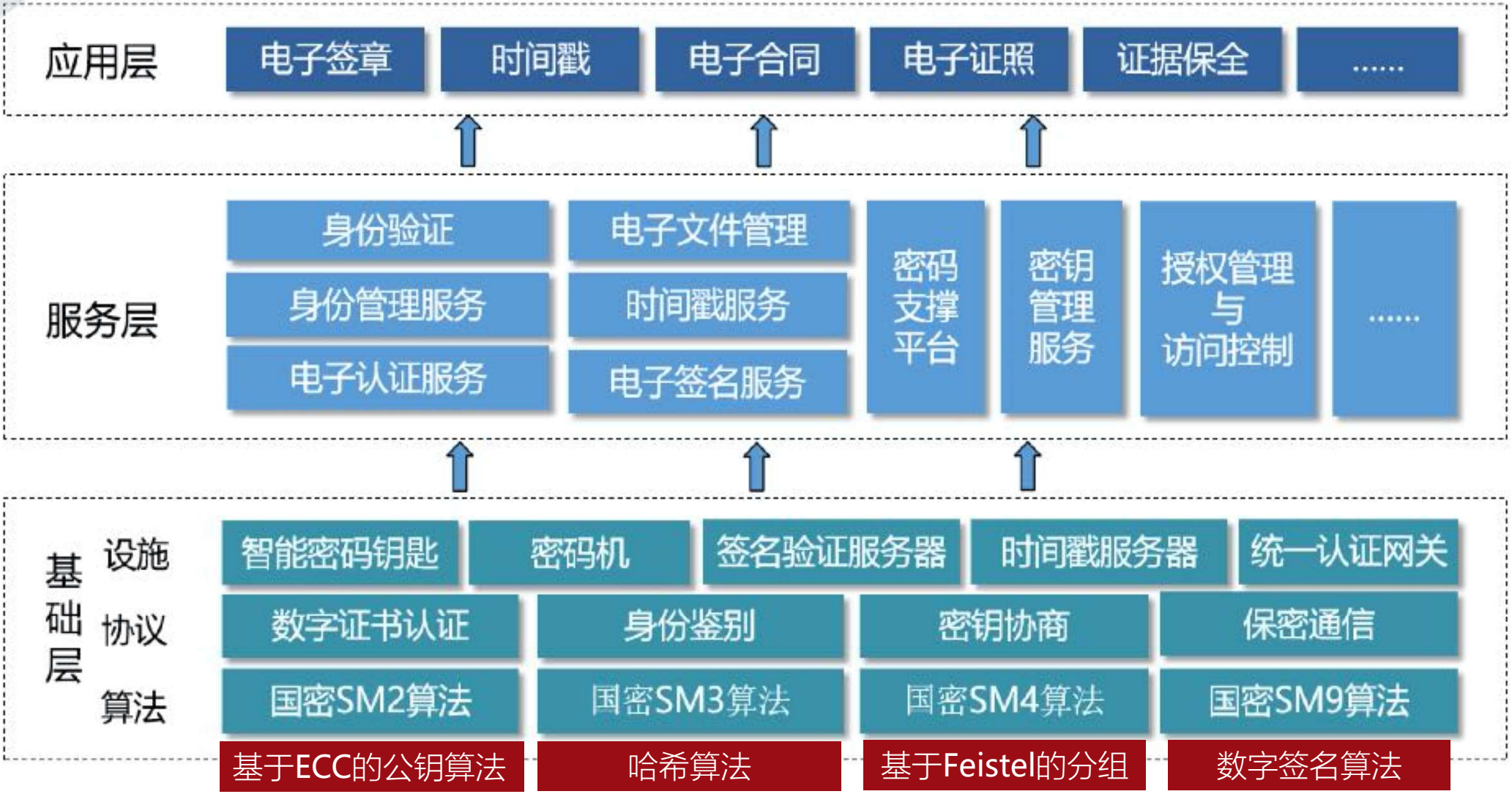

Any Questions?

总结

总结


信息安全
三大目标CIA
机密性
完整性
可用性
三个概念AAA
保障性
真实性
匿名性
常见安全威胁与攻击
窃听
改造
伪装
相关性追溯
拒绝服务
十大安全原则
密码学：保障信息安全的最基础工具
按时间划分：标志——1949年香农发表保密系统的通信理论
古典密码：依赖于加密算法的精心设计和保密性，以手工密码和机械密码为主
移位密码：斯巴达棒、凯撒密码、维吉尼亚密码
置换密码：栅栏密码
现代密码：公开了加密所用的算法，注重的是对密钥的保密，以电子密码和计算机密码为主——密码学成为一门学科
扩散：让明文中的每一位影响密文中的许多位
仿射密码、代换操作与混淆：将密文与密钥之间的统计关系变得尽可能复杂
按密钥形式划分：
流密码
分组密码：5大模式
ECB
CBC
CFB
OFB
CTR
按加密方式划分
对称加密
DES：56位密钥64位分组明文——基于Feistel结构
初始置换（IP）
分为左右两半
16轮Feistel函数——仅对右半运行
逆初始置换（IP-1）
扩展置换E
轮密钥异或——密钥产生算法
选择置换1（PC-1）
循环位移
选择置换2（PC-2）
混淆代换S盒（8个）
扩散置换P盒（1个）
与左半异或
左右交换进入下一轮Feistel
AES：128/192/256位密钥128位分组明文——基于SPN结构
轮密钥异或——密钥扩展算法
9轮SPN结构
字节代换S盒
行位移
列混淆
轮密钥异或
第10轮SPN结构（思考：为什么不需要列混淆）
字节代换S盒
行位移
轮密钥异或
非对称加密
DH密钥交换协议
RSA——基于质数分解难题
ECC——基于椭圆曲线的离散对数难题
基础应用：
数字签名：私钥签名公钥验证
MAC：附加验证标记

应用


应用层
电子签章
时间戳
电子合同
电子证照
证据保全
……
服务层
身份验证
身份管理服务
电子认证服务
电子文件管理
时间戳服务
电子签名服务
密码支撑平台
密钥管理服务
授权管理与访问控制
……
基础层
设施
协议
算法
智能密码钥匙
密码机
签名验证服务器
时间戳服务器
统一认证网关
数字证书认证
身份鉴别
密钥协商
保密通信
国密SM2算法
国密SM3算法
国密SM4算法
国密SM9算法
基于ECC的公钥算法
哈希算法
基于Feistel的分组
数字签名算法

- 用户和系统数据保护

  地铁系统中有大量的数据传输，包括乘客信息、运营数据、监控视频等。使用如AES这样的分组密码算法可以保证这些数据在传输和存储过程中的安全性和保密性。

- 用户和设备身份验证授权

  系统需要对员工和设备进行身份验证，确保进入系统的员工身份和设备来源的可靠性，RSA和其他非对称加密技术可以用来安全地验证用户和设备身份，并确保只有授权的用户才能访问敏感的系统资源，保障信号设备和相关组件的软件更新和数据传输是可信的。。

- 应用场景案例
  - 实时数据通信安全

    地铁系统的运行依赖于实时数据的准确和安全的传输。非对称加密技术可以用来安全地交换密钥，之后可以使用这些密钥进行更高效的对称加密，保证通信的安全。
  - 系统数据流监控

    对内网传输的重要数据加密，确保关键操作和环境监控信息的安全，避免数据泄露或被篡改。
  - 完整性检查

    结合哈希算法、对称加密等方式对数据进行校验，确保数据的完整性，避免数据因网络波动、恶意攻击等原因损坏进而导致故障。

资料推荐

书籍推荐

- 《密码简史》
- 杨义先、钮心忻著

课程推荐

Stanford University

斯坦福 Cryptography I

https://www.coursera.org/learn/crypto

前沿研究

量子密码
DNA密码
混沌密码
基于格的密码
多变量二次方程密码