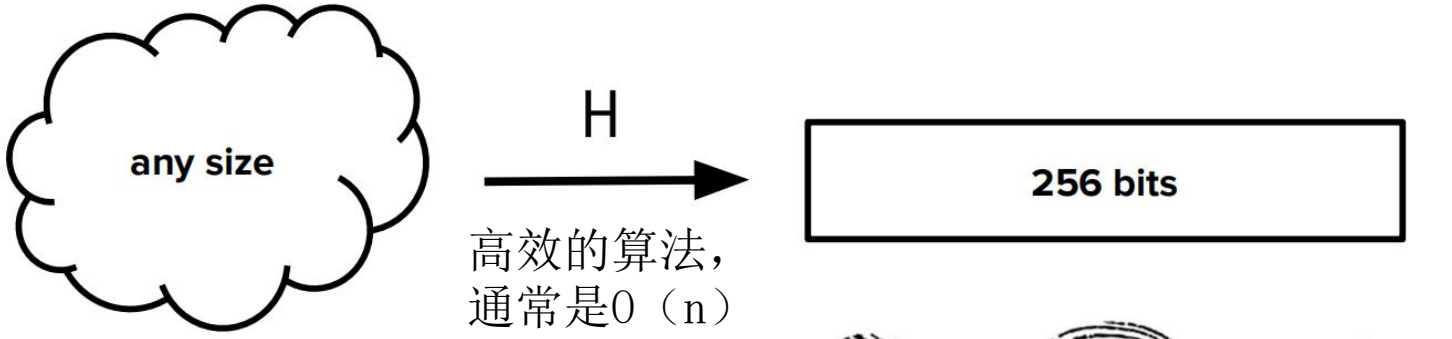
第五章　密码学基础

目录

哈希（hash）

1.1 Hash函数定义

Hash函数又称为散列函数或哈希函数，是一种从**任何种类、任何大小的数据中创建固定大小值**的方法，这个值也称为散列值（Hash值），通常用一个短的随机字母和数字组成的字符串来代表。



高效的算法，
通常是O（n）

通常散列值在应用中被称为指纹（fingerprint）或摘要（digest）。

1.2 Hash函数特征

1. **逆向难解。**根据给定的散列值，以目前的计算资源很难在有限的时间内逆向推出原文。

2. **强抗碰撞性。**对于任意两个不同的明文x和y，H（x）不等于H（y）。

3. **雪崩效应。**原文的微小变更会导致散列值巨大的变化，使得通过明文的细微变化推导出散列值或者散列值的微小变化推导出明文变得非常困难。

```
I am Satoshi Nakamoto0 => a80a81401765c8eddee25df36728d732...
I am Satoshi Nakamoto1 => f7bc9a6304a4647bb41241a677b5345f...
```

1.3 Hash函数常见算法

MD（Message Digest）

MD4（RFC 1320）是MIT的Ronald L. Rivest在1990年设计的，其输出为128位。MD4已被证明不够安全。
MD5（RFC 1321）是Rivest于1991年对MD4的改进版本，其输出是128位，MD5比MD4更加安全，但计算速度较慢。

SHA（Secure Hash Algorithm）

SHA是一个Hash函数族，NIST于1993年发布其首个实现。主要有SHA-0、SHA-1、SHA-2和SHA-3三个版本。SHA-2版本包括SHA-224、SHA-256、SHA-384和SHA-512，其中SHA-256是应用最广泛的Hash函数。

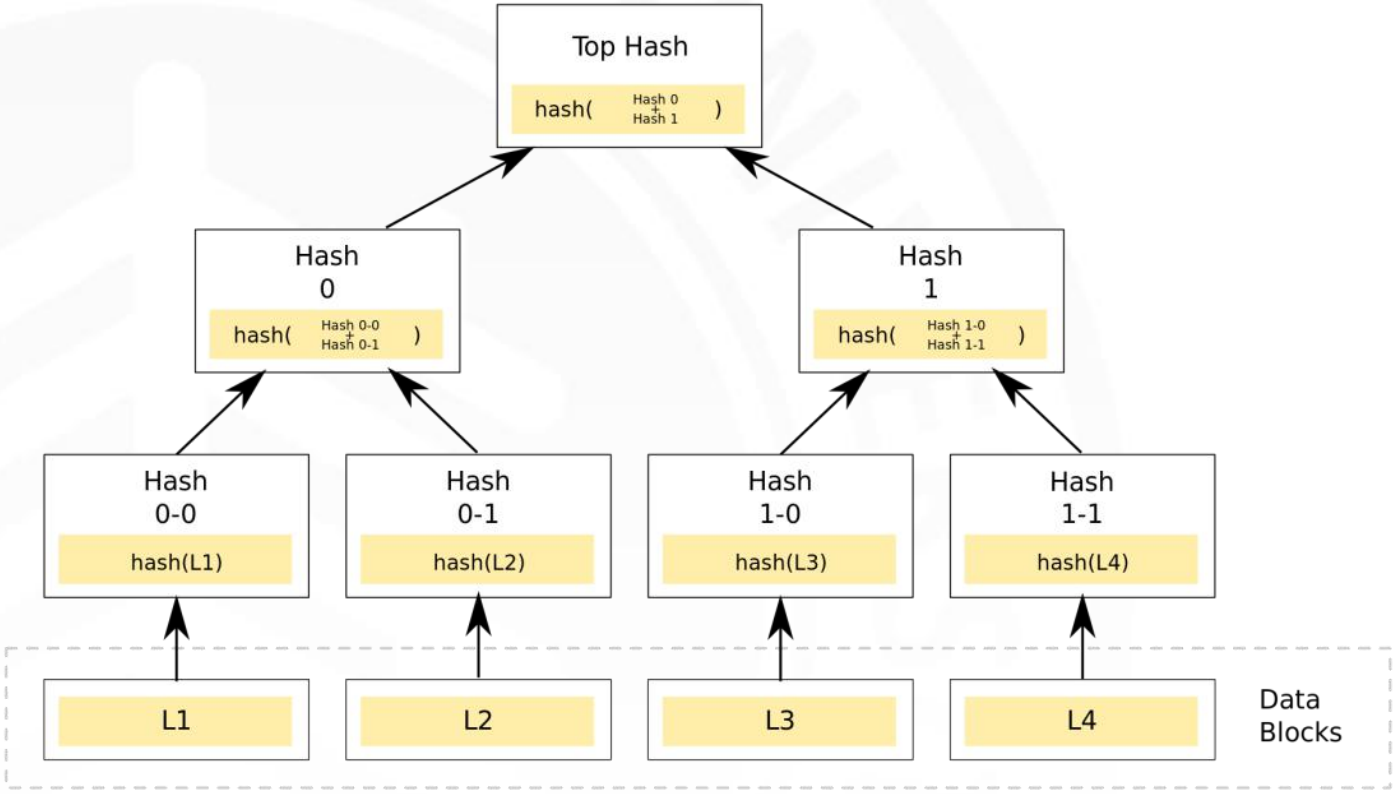
MD5、SHA-0和SHA-1已经发现碰撞，存在安全问题，已经不适应商业应用。

1.4 Hash函数在区块链中的应用

1. **区块链的地址与标识。**每个交易的标识、区块的标识都是一个Hash值。区块链通过区块链头中的PrevBlockHash或ParentHash连接相邻的块，最终形成链状结构。

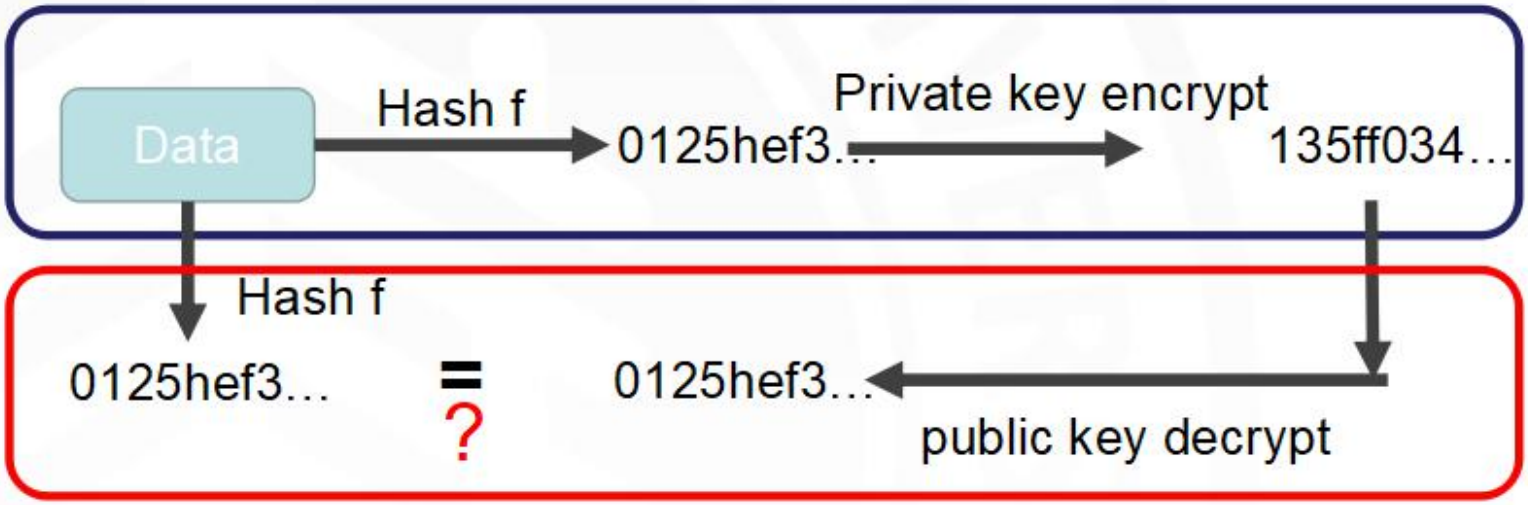
2. **区块体相关数据的树根hash－交易树。**在区块的头部存储Merkle树的根散列值，这个根散列值是由两两交易逐层生成的，因此这个根散列值也代表了区块中所有交易数据的数字摘要。

1.4 Hash函数在区块链中的应用



1.4 Hash函数在区块链中的应用

3. 签名与验证



1.4 Hash函数在区块链中的应用

4. **防篡改。**区块中某个交易的修改会导致交易的散列值发生变化，区块头部中的根散列值也发生变化。区块的校验方法会校验区块头部中的根散列值和区块中交易生成的新的根散列值是否一致。

5. **数据编码。**K-V序列对中key值的Hash。

加密解密算法

2.1 现代加解密算法组件

1. **加解密算法。**加解密算法是密码学核心技术，自身固定不变，一般公开可见。

2. **加解密密钥。**密钥是关键信息，需要进行特殊保护。一般来说，对同一种算法，密钥越长，加密强度越大。

区块链系统中往往使用经过长时间各方面充分实践论证的算法，而不是自行设计和发明不公开的未经充分验证的新算法。

2.2 攻击类型

1. **唯密文攻击**。指的是在仅已知加密文字（即密文）的情况下进行攻击，比如基于密文上的统计分析。
2. **已知明文攻击**，假设攻击者能够获取部分明文和相应密文，如截取信息前段，通过该类型攻击获取加密方式，从而便于破解后段密文。
3. **选择明文攻击**，指的是攻击者拥有加密机的访问权限，可构造任意明文所对应的密文。其目的是获得解密用的钥匙。
4. **选择密文攻击**，指的是攻击者掌握对解密机的访问权限，可构造任意密文所对应的明文。透过未知的密钥获得解密后的明文。

四种攻击模型，攻击强度依次增大，对密码算法和密钥分发提出更大的安全性要求。

2.3 加解密算法分类

2.4 对称加密

▲ **特点：**加解密的密钥相同。

▲ **优点：**一般加解密效率和加密强度都很高。

△ **缺点：**参与方都持有密钥，有人泄露则安全性被破坏。密钥分发也存在问题，需要借助 Diffie-Hellman 协议或非对称加密方式来实现。

2.4 对称加密 -- 分类

- **分组密码**：将明文切分为定长数据块作为基本加密单位。主要有**DES、3DES、AES、IDEA等**。DES和3DES目前已经认为不够安全；AES处理速度快，目前尚未有有效的破解手段；IDEA设计类似于3DES，具有更好的加密强度。

- **序列密码**：也称为流密码，每次对一个字节或字符进行加密处理。

2.5 非对称加密

特点：加密密钥和解密秘钥是不同的，分别称为公钥（public key）和私钥（private key）。私钥一般需要通过随机数算法生成，公钥可以根据私钥生成。公钥一般公开，他人可获取。私钥一般个人持有，他人不能获取。其安全性往往需要基于数学问题来保障。

优点：公私钥分开，可以在不安全通道中使用。

缺点：处理速度比较慢，一般比对称加密算法慢2~3个数量级，同时加密强度也往往不如对称加密算法。一般用于签名或密钥协商，但不适合大量数据的加解密。

2.5 非对称加密 -- 分类

非对称加密代表算法包括：**RSA、ELGamal、椭圆曲线等系列算法**。RSA的安全性基于及大整数因数分解的难度，目前没有可靠的攻击方式；ELGamal的安全性基于循环群上的离散对数难题；椭圆曲线算法安全性更高，开销更大。商业上一般推荐椭圆曲线系列算法。

2.6 混合加密机制

- 在实际应用中用非对称加密算法协商一个公共密钥，然后用公共密钥对数据进行加解密。

2.7 椭圆曲线（ECC）-- 算法

1. 选定有限域椭圆曲线$EP(a,b)$，p为质数，$x,y \in [0,p-1]$：
$$y^2 = x^3 + ax + b \pmod p$$

2. 选择两个满足下列约束条件的小于p的非负整数a,b：
$$4a^3 + 27b^2 \neq 0 \pmod p$$

3. 选定基点G，G为椭圆曲线$EP(a,b)$上的点。

4. 确定G的阶n，$n * G = O\infty$。

5. 根据n产生非对称密钥：私有密钥k $(k < n)$，并生成公开密钥$K = kG$。

2.7 椭圆曲线（ECC）-- 相关定义

1. **椭圆曲线。**一条椭圆曲线是在**射影平面**上满足**威尔斯特拉斯方程**（Weierstrass）所有点的集合。椭圆曲线的形状并不是椭圆的，只是因为椭圆曲线的描述方程，类似于计算一个椭圆周长的方程故得名。

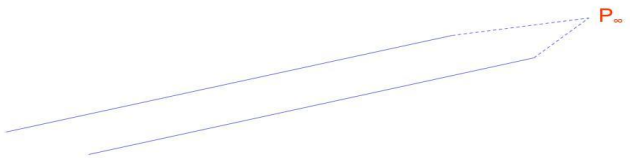
$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$$

2. **射影平面。**射影平面是普通平面上全体**无穷远点**与全体普通平面的点构成的射影平面。普通平面上点(x, y)，令$x = X/Z$，$y = Y/Z$，$Z \neq 0$，则投影为射影平面上的点$(X:Y:Z)$。

▲ 例如求普通平面点（1,2）在新的射影平面坐标体系下的坐标：（1:2:1）（2:4:2）（1.2:2.4:1.2）等形如（Z:2Z:Z），Z≠0的坐标都是（1,2）在射影平面上的点。

2.7 椭圆曲线（ECC）-- 相关定义

3. **无穷远点0∞。**一条直线只有一个无穷远点，一对平行线有公共的无穷远点。任何两条不平行的直线有不同的无穷远点（否则会造成有两个交点）。平面上全体无穷远点构成一条无穷远直线。



4. **维尔斯特拉斯方程。**维尔斯特拉斯标准方程形式（Weierstrass normal form）：$y^2 = x^3 + ax + b$
这个限定条件是为了保证曲线不包含**奇点**(singularities)： $4a^3 + 27b^2 \neq 0$

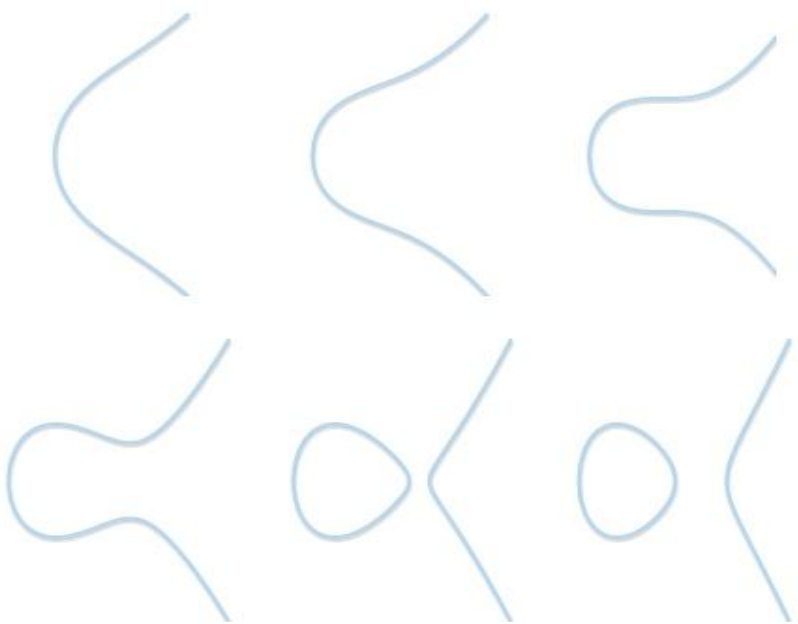
2.7 椭圆曲线（ECC）-- 相关定义

5. **奇异点。**左图，带锐点(式1)；右图，曲线自交（式2）。他们都不是有效的椭圆曲线

式1: $y^2 = x^3$

式2: $y^2 = x^3 - 3x + 2$

2.7 椭圆曲线（ECC）-- 示例



不同的椭圆曲线对应不同的形状
（b=1，a从2到-3）

随着a和b的不同，椭圆曲线也会在平面上呈现出不同的形状，但他还是很容易辨认的，椭圆曲线始终是关于x轴对称的。

2.8 有限域椭圆曲线

椭圆曲线是连续的，并不适合用于加密；所以，必须把椭圆曲线变成**离散**的点，形成定义在**有限域上的椭圆曲线**。引入**阿贝尔群**(abelian group)（满足交换律），对群中元素进行二元运算，如加法运算。

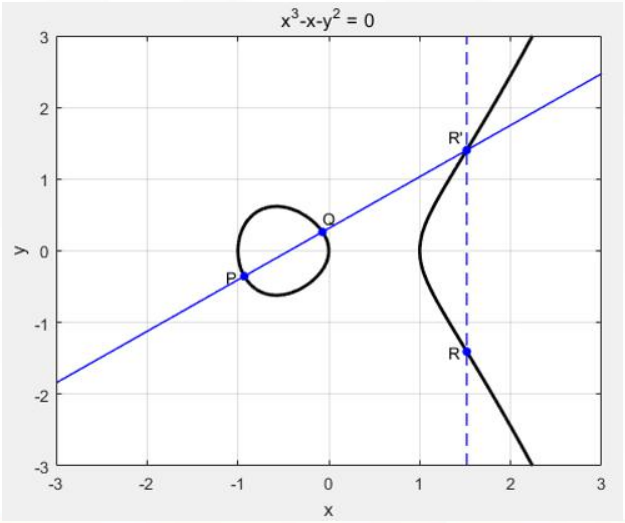
群是一种代数结构，由一个集合以及一个二元运算所组成。已知集合和运算$(G, +)$，群必须满足如下要求：

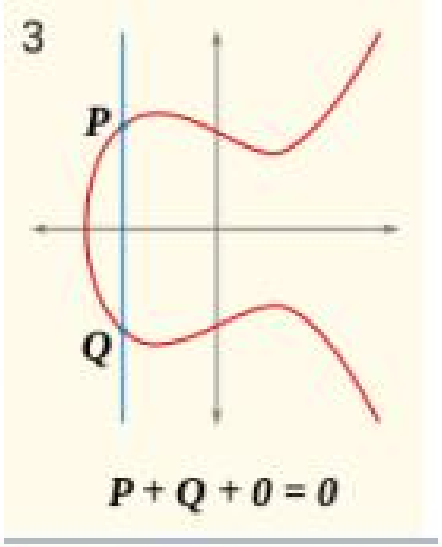
- **封闭性**（closure）：$\forall a, b \in G$，$a + b \in G$。
- **结合律**（ associativity ）：$\forall a, b, c \in G$，有$(a + b) + c = a + (b + c)$。
- 存在一个**单位元**（identity element）：$\exists e \in G$，$\forall a \in G$，有$e + a = a + e = a$。
- 每个数都存在一个**相反数**(inverse)：$\forall a \in G$，$\exists b \in G$ 使得$a + b = b + a = e$。
- **交换律**(commutativity)：$\forall a, b \in G$，$a + b = b + a$。

2.8 有限域椭圆曲线 -- 算法

1. 在椭圆曲线上定义一个群G，群中的元素就是椭圆曲线上的点，单位元就是无穷远点$O\infty$，相反数P，是关于X轴对称的另一边的点。

2. 定义群的 $+$ 运算子。取E上的两点P, Q，若两者相异，$P + Q$表示穿过P和Q的弦和椭圆曲线相交的第三点，再经x轴反射的镜像点；若P和Q的弦与y轴平行，$P + Q = 0$（无限远点）。

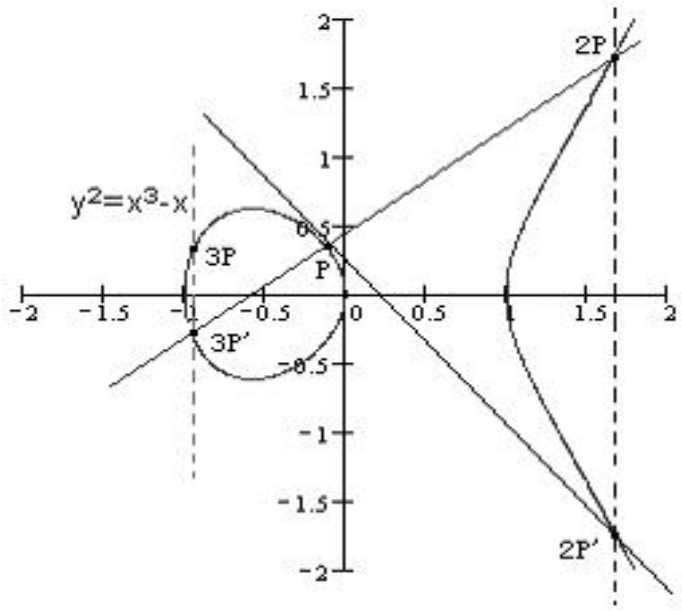




2.8 有限域椭圆曲线 -- 算法

3. 定义 $+$ 运算子。若两者是同一点，$P+P=2P$表示以P为切点和椭圆曲线相交的点再经x轴反射的镜像点。

4. 定义第二个运算“$*$”。定义$2*P=P+P$, $3*P=2*P+P=P+P+P$等等。注意给定整数j和k， $j*(k*P)=(j*k)*P=k*(j*P)$



2.8 有限域椭圆曲线 -- 算法

5. **有限域椭圆曲线点的阶**。如果椭圆曲线上一点P，存在最小的正整数n使得$n * P = O\infty$，则将n称为P的阶。若n不存在，则P是无限阶的（可以理解为$P + P + ... + P = O\infty$）。

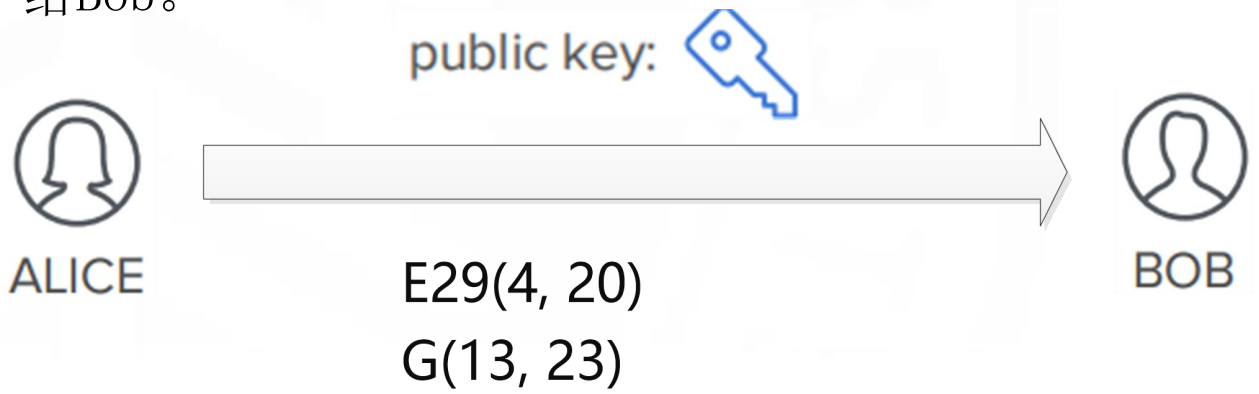
6. 椭圆曲线离散对数问题（ECDLP）就是给定点P和Q，确定整数k使$k * P = Q$，但这被广泛承认求解k是很困难的。而椭圆曲线加密就依赖求解该问题的困难性上。$K = k * G$，$C1 = r * G$。我们称它为“对数”而不是“除”是为了和其他密码系统保持一致性（用幂来代替乘）。

2.8 有限域椭圆曲线 -- 加密过程

1. Alice选定一条椭圆曲线E：$E29(4,20)$，基点$G(13,23)$，基点G的阶数$n = 37$。

2. Alice选择一个私有密钥k（$k < n$）比如25，并生成公开密钥$K = kG$，$K = kG = 25G = (14,6)$。

3. Alice将E（椭圆曲线）和点K（公钥）、G（基点）传给Bob。



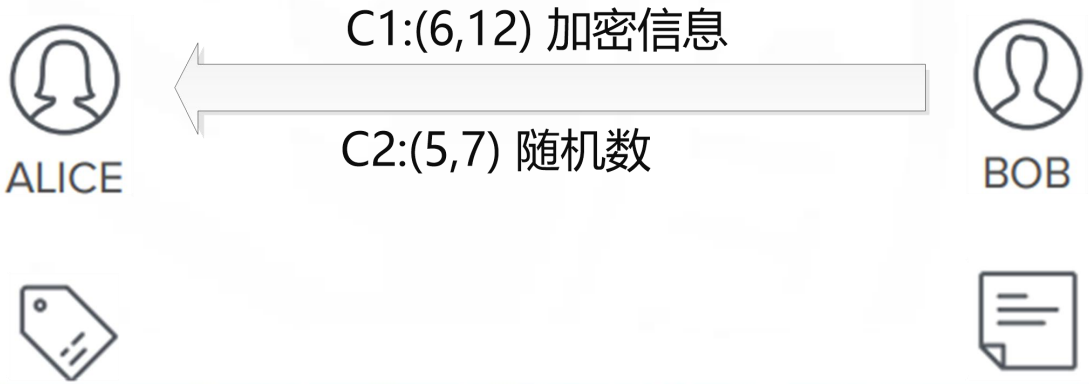
2.8 有限域椭圆曲线 -- 加密过程

4. Bob收到信息后，将待传输的明文编码到椭圆曲线上的一点M（编码方法略），并产生一个随机整数r（$r < n$, n为G的阶数）。假设$r = 6$，$M = (3,28)$。

5. Bob计算点$C1 = M + rK$和$C2 = rG$。

- $C1 =$ 用公钥+随机数加密后的加密信息 $M + r * k * G$，
  $M + 6K = M + 6 * 25 * G = M + 2G = (3,28) + (27,27) = (6,12)$。
- $C2 =$ 选择的随机数 $6G = (5,7)$。

6. Bob将$C1$、$C2$传给Alice。



2.8 有限域椭圆曲线 -- 加密过程

7. Alice收到信息后，计算$C1-kC2$，结果就应该是点M。

- $M=C1-kC2=(6,12)-25C2=(6,12)-25*6G$
  $=(6,12)-2G=(6,12)-(27,27)=(6,12)+(27,2)=(3,28)$
- $M=C1-kC2=M+r*k*G-k*r*G$
- $K=kG$
- $C2=rG$，$kC2=krG=rK$，即 $C1-kC2=C1-rK$，r未知
- k是私钥
- 需要破解出M，就需要由$C2=rG$求解出r，然后根据r，K和$C1$来求解M，但由$C2=rG$求解出r是一个椭圆曲线离散对数问题（ECDLP）。

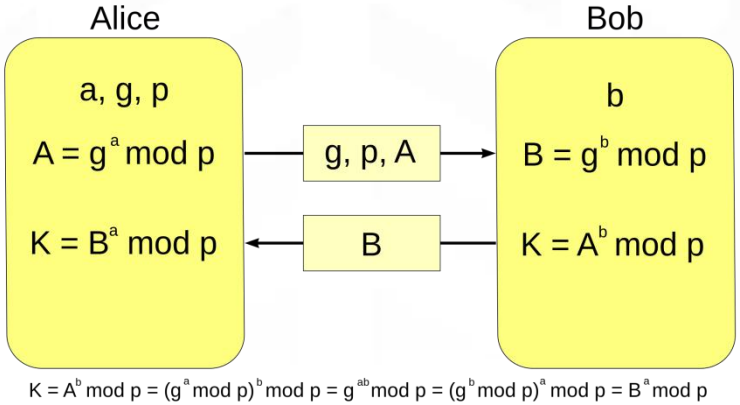
2.9 Diffie–Hellman密钥交换（基于数论的模运算的方案）

- Diffie－Hellman密钥交换是基于离散对数问题通过公共信道交换一个信息，就可以创建一个可以用于在公共信道上安全通信的共享秘密（shared secret）。即 $g^{ab} \bmod p = g^{ba} \bmod p$。 算法中$a$， b 和 $g^{ab} \bmod p$（秘钥k）、 $g^{ba} \bmod p$（秘钥k）是秘密的。 其他所有的值p, g, $g^a \bmod p$以及 $g^b \bmod p$都可以在公共信道上传递。

- 在实际应用中为了更好的安全性，必须使用非常大的 a, b 以及 p，否则如例子中 $g^{ab} \bmod 23$的可能取值(总共有最多22个这样的值，就算a和b很大也无济于事)。

2.9 Diffie–Hellman密钥交换 -- 算法过程

1. Alice和Bob协商确定p以及原根g，如：$p = 23$，$g = 5$。
2. Alice选择一个密码数字$a = 6$，计算$A = g^a \bmod p$并发给Bob。
3. Bob选择一个密码数字$b = 15$，计算$B = g^b \bmod p$并发给Alice。
4. Alice计算$K = Ba \bmod p$。
5. Bob计算$K = Ab \bmod p$。



$$K = A^b \bmod p = \left(g^a \bmod p\right)^b \bmod p = g^{ab} \bmod p$$

$$g^{ab} \bmod p = \left(g^b \bmod p\right)^a \bmod p = K'$$

$$K = K'$$

Alice和Bob最终都得到了同样的值K，也就是对称密钥。

2.9 Diffie–Hellman密钥交换 -- 攻击方法

- 攻击者为了获知K，需要根据g和$A = g^a \bmod p$，计算出a，然后用类似方法计算出b，进而求解出K，但该问题是一个离散对数问题没有有效方法求解，因此可以保证算法的安全性。

数字签名

3.1 数字签名概念

- **定义：**数字签名又称为公钥数字签名（Digital Signature），是基于非对称加密机制实现消息内容的加密与验证。

- **特性：**完整性（消息在传输过程中没有丢位，没被篡改），认证性（消息的发送者是发布公钥的Alice），不可否认性（Alice的确发布过该消息）。

- **组成：**包括合法机构生成的公钥和私钥，一个签名算法（给定消息内容和私钥后，生成签名），一个签名验证算法（给定消息内容、公钥和签名后，验证消息内容的真实性）。

3.2 签名过程

1. Alice与Bob互换公钥。
2. Alice对消息内容进行摘要（如内容的Hash值），然后用自己的私钥对摘要进行加密（签名）。
3. Alice用Bob的公钥对数字签名和消息内容进行加密并发送给Bob。
4. Bob接收到消息后，用Bob的私钥进行解密，得到消息内容和Alice的数字签名
5. Bob用Alice的公钥对数字签名进行解密得到数字摘要。
6. Bob对消息内容进行数字摘要，并与解密的数字摘要进行比对。

如果数字摘要一致，则说明消息确实是从Alice发送过来并且消息内容未被篡改过。

3.3 签名分类

3.4 盲签名（blind signature）

- 消息的内容在签名之前对签名者是不可见的（盲化）。
- 得到的盲签名可以对原始的、非盲消息以常规数字签名的方式公开验证。
- 盲签名可以有效地保护隐私，其中签名者和消息作者是不同，例子包括电子选举和数字现金。

3.5 多重签名（multiple signature）

- 多重签名即n个签名者中，收集到至少m个（$n \geq m \geq 1$）签名，即认为合法。其中，n是提供的公钥个数，m是需要匹配公钥的最小签名个数。
- 在加密货币领域，多重签名可以简单的理解为一个数字资产的多个签名。多重签名预示着数字资产可由多人支配和管理。如果要动用一个加密货币地址的资金，通常需要该地址的所有人使用他的私钥（由用户专属保护）进行签名。
- 一个多重签名地址可以关联n个私钥，在需要转账等操作时，只要其中的m个私钥签名就可以把资金转移了，其中m要小于等于n，也就是说m/n小于1，可以是2/3，3/5等等，是要在建立这个多重签名地址的时候确定好的。

3.6 群签名（group signatures）

- 群签名方案允许用户代表群签名消息，并在该群内保持匿名。看到签名的人可以用公钥验证该消息是由该群成员发送的，但不知道是哪一个。

- 用户不能滥用这种匿名行为，因为群管理员可以通过使用秘密信息(密钥)来消除（恶意）用户的匿名性。例如，大公司里的雇员可以使用群签名方案对消息进行签名，其中验证者知道消息是由他们公司里的雇员签名的就足够了，不需要知道是由哪个特定雇员签名的。

- 群签名方案的关键是“群管理员”，它负责添加群成员，并能够在发生争议时揭示签名者身份。在一些系统中，添加成员和撤销签名匿名性的责任被分开，并分别赋予给群管理员和撤销管理员。

3.6 群签名 -- 签名过程

1. **初始化。**群管理者建立群资源，生成对应的群公钥（Group Public Key）和群私钥（Group Private Key）群公钥对整个系统中的所有用户公开，比如群成员、验证者等。

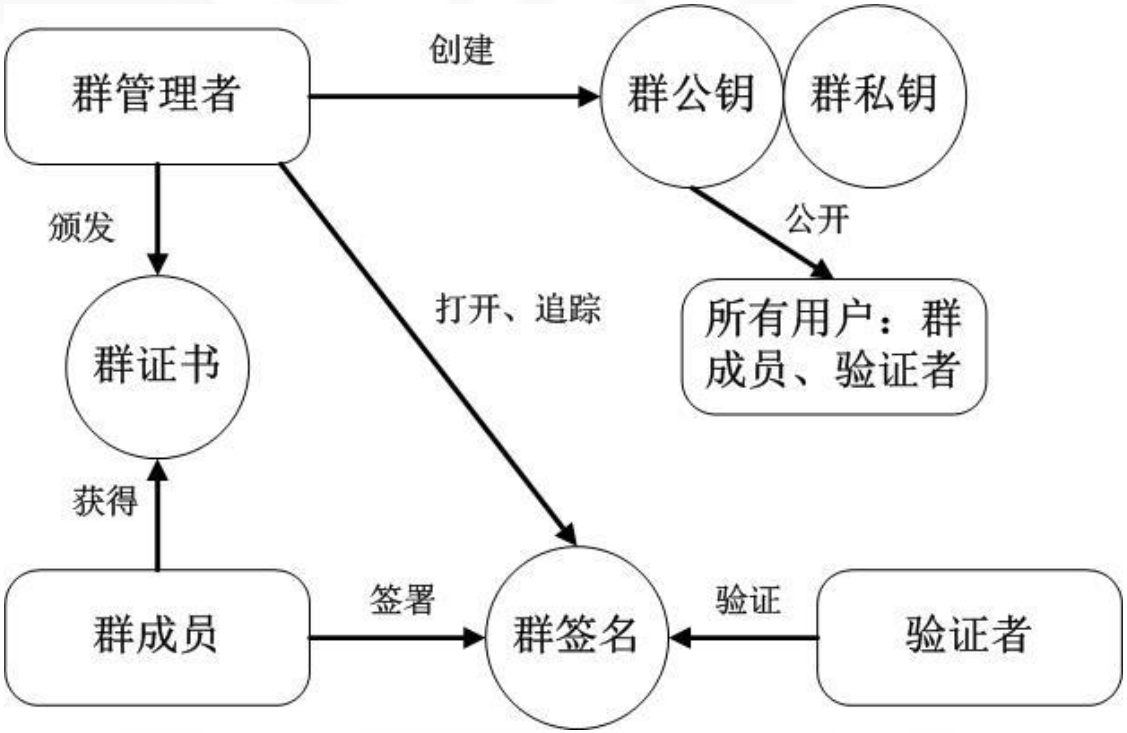
2. **成员加入。**在用户加入群的时候，群管理者颁发群证书（Group Certificate）给群成员。

3. **签名。**群成员利用获得的群证书签署文件，生成群签名。

4. **验证。**基于证书信任链，可以通过群公钥验证群所颁发的证书，同时验证者利用群公钥仅可以验证所得群签名的正确性，但不能确定群中的正式签署者。

5. **打开。**群管理者利用群私钥可以对群用户生成的群签名进行追踪，并暴露签署者身份。

3.6 群签名 -- 签名过程



3.7 环签名

- 环签名是一种简化的群签名,环签名中只有环成员没有管理者,不需要环成员间的合作。在环签名中不需要创建环，改变或者删除环，也不需要分配指定的密钥，无法撤销签名者的匿名性，除非签名者自己想暴露身份。

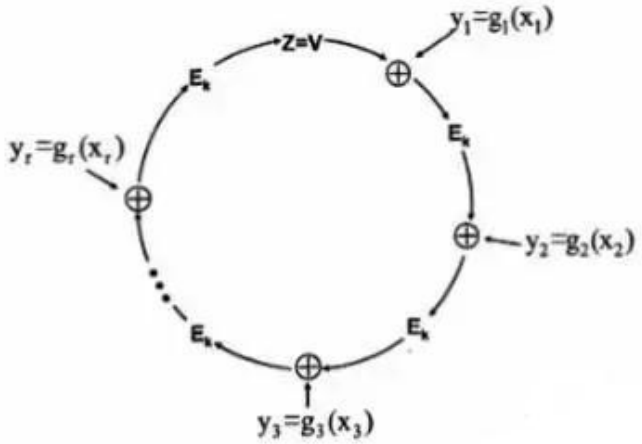
- 环签名方案中签名者首先选定一个临时的签名者集合，集合中包括签名者。然后签名者利用自己的私钥和签名集合中其他人的公钥就可以独立的产生签名，而无需他人的帮助。签名者集合中的成员可能并不知道自己被包含在其中。

3.7 环签名 -- 算法

1. 假定有n个用户，每一个用户拥有一个公钥和与之对应的私钥。

2. 密钥生成：为除签名者外的$n-1$个用户生成密钥对(公钥PKi, 私钥SKi)。

3. 签名：签名者用自己的私钥和n个环成员的公钥（$PK1, PK2, ..., PKn$）对消息m进行签名产生一个签名R，其中R中的某个参数根据一定的规则呈环状。



4. 签名验证：根据消息m和环签名R，按照一定环签名规则验证R是否为m的环签名

3.7 环签名 -- 安全性

- **无条件匿名性。**攻击者无法确定签名是由环中哪个成员生成，即使在获得环成员私钥的情况下,概率也不超过$1/n$。[签名者隐藏在环成员之中]。

- **正确性。**签名必需能被所有其他人验证。

- **不可伪造性。**环中其他成员不能伪造真实签名者签名, 外部攻击者即使在获得某个有效环签名的基础上, 也不能为消息m伪造一个签名。

数字证书及PKI体系

4.1 数字证书概念

- 数字证书（digital certificate）又称为公开密钥认证（Public key certificate），是一种证明公开密钥拥有者身份的电子文件，实现用户公钥的安全分发。
- 数字证书包含了公钥信息、拥有者身份信息（主体，subject）、以及数字证书认证机构（发行者，issuer）对这份文件的数字签名，以保证这个文件的整体内容正确无误。
- 证书拥有者凭此电子文件可向相应系统或其他用户表明身份，从而获得对方信任并可授权访问或使用某些计算资源或服务。
- 数字证书一般由第三方数字证书认证机构（CA，Certificate Authority）颁发，也可以由用户创建测试用或机构内部使用的数字证书，但这种证书不会被第三方系统或用户接受，仅仅用于测试或者机构内部使用。

4.2 数字证书格式

数字证书的格式普遍采用的是X.509 V3国际标准，包含以下一些内容：

证书的版本信息。
证书的序列号，每个证书都有一个唯一的证书序列号。
证书所使用的签名算法，如sha256WithRSAEncryption或ecdsa-with-SHA256。
证书的发行机构名称，命名规则一般采用X.500格式，如“C=CN, ST=Beijing, O=org.issuer.com, CN=ca.org.issuer.com”。
证书的有效期，通用的证书一般采用UTC时间格式，它的计时范围为1950-2049。
证书所有人的名称，命名规则一般采用X.500格式，如“C=CN, ST=Beijing, O=org.subject.com, CN=person.org.subject.com”。
证书所有人的公开密钥，包括公钥算法和公钥内容。
证书发行者对证书的签名。

数字证书可以二进制或Base64形式存储，常见的二进制文件扩展名有 .cer、.crt、.der，Base64的文件扩展名则通常是 .pem。如果把证书和私钥一起存储，则可以使用PKCS#12（.p12）格式。

4.3 数字证书种类

用户自己生成数字证书，没有任何可信赖的人签名，通常不会被广泛信任，用于小范围测试等。

自签证书（Self-signed certificate）

根证书（Root certificate）

一种自签证书，但获得广泛认可，通常已预先安装在各种软件，作为信任链的起点。通常来自于公认可靠的政府机关、软件公司等。

用于签发其他证书的证书，该证书必须被上层中间证书或者根证书签名。

中间证书（Intermediate certificate）

实体证书（End-entity or leaf certificate）

不能用于签发其他证书的证书，例如TLS / SSL服务器和客户端证书，电子邮件证书，代码签名证书。

4.5 PKI体系

- PKI（Public Key Infrastructure），也称为公钥基础结构，用于创建、存储和分发数字证书的系统，用于验证特定公钥是否属于特定实体。
- PKI 创建数字证书，将公钥映射到实体，安全地将这些证书存储在中心数据库中，并在需要时将其撤销。
- PKI体系一般由政府机构或者可信任的第三方机构建立和维护。

4.5 PKI体系 -- 组件

1. **证书颁发机构（Certificate Authority，CA）：**接收来自RA的请求，负责存储、颁发和签名数字证书。

2. **登记机构（Registration Authority，RA）：**对用户身份进行验证，校验身份信息的合法性并登记，审核通过发送给CA，由CA颁发该用户实体的数字证书。

3. **证书存储中心：**存储证书数据以及证书索引。

4. **证书管理系统（Certificate Management System）：**提供访问或传递已颁发证书的功能。

05 国密算法

5.1 国密算法

- 国产密码算法（国密算法）是指国家密码局认定的国产商用密码算法。
- 在金融领域目前主要使用公开的SM2、SM3、SM4三类算法，分别是非对称算法、哈希算法和对称算法。

5.2 国密算法 -- SM2

- SM2椭圆曲线公钥密码算法是我国自主设计的公钥密码算法。
- SM2包括SM2-1椭圆曲线数字签名算法，SM2-2椭圆曲线密钥交换协议，SM2-3椭圆曲线公钥加密算法，分别用于实现数字签名密钥协商和数据加密等功能。
- SM2算法是基于椭圆曲线上点群离散对数难题，相对于RSA算法，256位的SM2密码强度已经比2048位的RSA密码强度要高。

5.3 国密算法 -- SM3

- SM3哈希算法是我国自主设计的密码杂凑算法，适用于商用密码应用中的数字签名和验证消息认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。

- SM3算法的输出长度为256比特，因此SM3算法的安全性要高于MD5算法（128比特）和SHA-1算法（160比特）。

5.4 国密算法 -- SM4

- SM4分组密码算法是我国自主设计的分组对称密码算法，用于实现数据的加密/解密运算，以保证数据和信息的机密性。

- SM4算法与AES算法具有相同的密钥长度分组长度128比特，安全性上高于3DES算法。

谢谢！