Web3 常用的密码算法

lynndell2010@gmail.com

1 对称加密

凯撒密码

已知最早的代替密码是由 Julius Caesar 发明的 Caesar。它非常简单，就是对字母表中的每个字母，用它之后的第三个字母来代替。例如：

明文：meet me after the toga party

密文：PHHW PH DIWHV WKH WRJD SDVWB

注意到字母表是循环的，即认为紧随 Z 后的是字母 A。我们通过列出所有的字母来定义如下变换：

明文：a b c d e f g h i j k l m n o p q r s t u v w x y z

密文：D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

① 当涉及字母时，本书约定：用小写字母表示明文，大写字母表示密文，斜体小写字母表示密钥。

如果我们让每个字母等价于一个数值：

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	o	p	q	r	s	t	u	v	w	x	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

那么加密算法可以如下表达。对每个明文字母 p，代替成密文字母 C[①]：

$$C = \mathrm{E}(3, p) = (p + 3) \bmod 26$$

移位可以是任意整数 k，这样就得到了一般的 Caesar 算法：

$$C = \mathrm{E}(k, p) = (p + k) \bmod 26 \qquad (2.1)$$

这里 k 的取值范围是从 1 到 25。解密算法是

$$p = \mathrm{D}(k, C) = (C - k) \bmod 26 \qquad (2.2)$$

对称加密

初始化：双方事先共享一个保密随机数 r，或使用相同的设备生成一个相同的随机数 r。

加密：发送方使用设备生成随机数 r，对消息 m，如下计算

$$c := r \oplus m$$

发送密文 c。

解密：接收方接收到密文 c，使用设备生成随机数 r，如下计算

$$m := r \oplus c$$

获得消息 m。

2RSA 非对称密码算法

欧几里得辗转相除法

假设 a≥b，求正整数 a 和 b 的最大公因数 gcd(a, b)?

如果 a mod b=0，则 gcd(a, b)=b；否则 gcd(a, b)=gcd(b, a mod b)，递归运算。

举例：

a=254, b=8;
254=8*h+6, h=31;
8=6*h+2,h=1;
6=2*3+0 余数为零，则最大公因子是 2
gcd(254,8)=gcd(8,6)=gcd(6,2)=2

多项式时间内可解的问题（简单问题、**P 问题**）
已知大合数 a，求小于 a 的大素数 b，使得 gcd(a,b)=1。
该问题能够使用**欧几里得辗转相除法**快速求解。

随机选择一个大素数 b，如果 $\gcd(a,b)\neq b$，则 gcd(a,b)=1。

因此，这样的大素数有很多，且很容易找。

困难问题
大数的因子分解问题，非常困难！
保密数据：随机选择 2 个保密大素数 p,q，计算 $n=p\cdot q$
公开： n。

问题：求 p,q？

举例 1：保密数据 5 和 7；公开数据 n=35
已知 35，求 p,q?
举例 2：保密数据 p,q；公开数据 n
公开 n=8239121061250502943937=p*q
8239121061250502943937 = [redacted]

For $2\le i\le\sqrt{n}$ **do**

For $2\le j\le\sqrt{n}$ **do**

If 8239121061250502943937= $i*j$

Break;
Break;
Output i, j.

如果是 2048bit 的大数 $n=2^{2048}-110+1$**，则计算机需要 10 年以上。**

但是，8 年抗日战争胜利以后，我就会公开 p 和 q。

NP 问题：求解时，需要 for 循环，暴力搜索，需要指数时间；
但是，一旦已知解，则可以快速验证解的正确性；

费马小定理
若 p 为素数，a 是正整数且不能被 p 整除，互素，则

$$a^{p-1}\equiv 1 \bmod p$$

举例 3：p=7，a=3，3 不能被 7 整除，则 $3^{7-1} \equiv 1 \bmod 7$；

$$3^6 \bmod 7 = (3*3)*(3*3)*(3*3) \bmod 7 = 9*9*9 \bmod 7 = 729 \bmod 7 = 1$$

举例 4：p=11，a=5，5 不能被 11 整除，则 $5^{11-1} \equiv 1 \bmod 11$；

$$5^{10} \bmod 11 = 9765625 \bmod 11 = 1$$

欧拉函数

小于 n 且与 n **互素**的正整数个数，记为 $\psi(n)$。习惯上 $\psi(1) = 1$

结论：如果 p 是素数，则 $\psi(p) = p - 1$ 例如：$\psi(37) = 36$

推论：如果有两个素数 p, q 且 $p \neq q$，那么对于 $n = p \cdot q$，有 $\psi(n) = (p-1) \cdot (q-1)$

$\psi(n) = \psi(q) \cdot \psi(p)$ 例如：$\psi(35) = \psi(5) \cdot \psi(7) = 4*6 = 24$

Determine $\phi(37)$ and $\phi(35)$.

Because 37 is prime, all of the positive integers from 1 through 36 are relatively prime to 37. Thus $\phi(37) = 36$.

To determine $\phi(35)$, we list all of the positive integers less than 35 that are relatively prime to it:

1, 2, 3, 4, 6, 8, 9, 11, 12, 13, 16, 17, 18
19, 22, 23, 24, 26, 27, 29, 31, 32, 33, 34

There are 24 numbers on the list, so $\phi(35) = 24$.

费马小定理：*若 p 为素数，a 是正整数且不能被 p 整除，则*

$$a^{p-1} \equiv 1 \bmod p$$

欧拉定理：对任意互素的 a 和 n，有 $a^{\psi(n)} \equiv 1$

因此，欧拉定理是费马小定理的一般化。
欧拉定理中：n 可以是合数，a 是素数与 n 互素也可以。

举例 5：n=7，a=3，互素，则 $3^{\psi(7)} \bmod 7 = 3^6 \bmod 7 = 729 \bmod 7 = 1$

$$3^{\psi(7)+1} \bmod 7 = 3^{6+1} \bmod 7 = 3^6 * 3^1 \bmod 7 = 1*3^1 = 3$$

$$3^{\psi(7)+\psi(7)+1} \bmod 7 = 3^{6+6+1} \bmod 7 = 729*729*3 \bmod 7 = 1*1*3 = 3$$

一般化 $3^{k \cdot \psi(7)+1} \bmod 7 = 3^1 \bmod 7 = 3, k = 1, 2, 3...$

一般化 $3^{k\cdot\psi(7)+r} \bmod 7 = 3^r \bmod 7, k = 1,2,3...$

结论：指数部分超过 $\psi(n)$，则是多余的，所以指数部分能够**模** $\psi(n)$

举例 8：n=7，a=3，互素，计算 $3^{601} \bmod 7$ 、$3^{1025} \bmod 7$

$$3^{601} \bmod 7 = 3^{100*\psi(7)+1} \bmod 7 = 3^{601 \bmod \psi(7)+1} \bmod 7 = 3^1 \bmod 7 = 3$$

$$3^{1025} \bmod 7 = 3^{170*6+5} \bmod 7 = 3^{170*\psi(7)+5} \bmod 7 = 3^5 \bmod 7 = 243 \bmod 7 = 5$$

欧拉定理：对任意互素的 a 和 n，有 $a^{\psi(n)} \equiv 1$

欧拉定理扩展：

$$a^{\psi(n)} \equiv 1$$
$$a^{i\cdot\psi(n)+1} \bmod n = a$$

以下计算是快速的（多项式时间内可计算）

对于 $n = p \cdot q$，已知 p, q，能快速计算 $\psi(n) = (p-1)\cdot(q-1)$

随机选择一个大素数 sk，使用欧几里得辗转相除法，快速测试与大合数 $\psi(n)$ 互素

$$\gcd(sk, \psi(n)) = 1$$

构造二元一次方程：$pk * sk - i * \psi(n) + 1 = 0$，有无数解。

计算下列循环：

For i in {1,2,…}.iterator {

$$pk := (i * \psi(n) + 1) / sk$$

}

Output: (pk, sk)

以下计算是慢速的（需要 for 循环，指数时间暴力搜索）

任务：公开 n, pk，不知道 $\psi(n)$，不能在多项式时间内计算计算 sk。

求解思路：已知 n，for 循环，在指数时间内计算 $\psi(n)$，需要 10 年

然后 For i in {1,2,…}.iterator {

$$sk := (i * \psi(n) + 1) / pk$$

}

Output: sk

$$pk \cdot sk = i * \psi(n) + 1$$

$$(m^{sk})^{pk} = (m^{pk})^{sk} = m^{pk \cdot sk} \bmod n = m^{i \cdot \psi(n)+1} \bmod n = m$$

应用场景 1：非对称的 RSA 公钥加密

已知：Alice 拥有公开数据是：pk 公钥；保密数据是：sk 私钥。

需求：Bob 需要将保密数据 m 发送给 Alice。

解决方案：

步骤 1：公钥对数据加密：Bob 计算 $C \leftarrow m^{pk} \bmod n$，将 C **广播**给 Alice

步骤 2：私钥对数据解密：Alice 接收 C，然后计算 $m \leftarrow (C)^{sk} \bmod n$，从 C 中计算出保密数据 m。

一致性过程如下：

$$(C)^{sk} \bmod n = (m^{pk} \bmod n)^{sk} \bmod n = m^{pk \cdot sk} \bmod n = m^{i \cdot \psi(n)+1} \bmod n = m$$

应用场景 2：数字签名

需求：Alice 公开宣称自己拥有一个公开交易单 m，

已知：Alice 拥有公开数据是：公钥 pk；保密数据是：私钥 sk。

解决方案：

步骤 1：Alice 计算：$\sigma \leftarrow m^{sk} \bmod n$，公开 (m, σ, pk)

步骤 2：**任可人**均可**校验**：$m == (\sigma)^{pk} \bmod n$

一致性过程如下：

$$(\sigma)^{pk} \bmod n = (m^{sk} \bmod n)^{pk} \bmod n = m^{sk \cdot pk} \bmod n = m^{i \cdot \psi(n)+1} \bmod n = m$$

应用场景 1 是 RSA 公钥加密，应用场景 2 是 RSA 数字签名。

RSA 这个密码算法，设计公钥密码算法，找这样的算法，RSA 设计上 100 个算法；R 找到从一本书里读到：一个数学家的日记：日历里写了一个大合数，全世界只有我知道这个大合数等于哪两个素数相乘。

举例 9：

1. Select two prime numbers, $p = 17$ and $q = 11$.
2. Calculate $n = pq = 17 \times 11 = 187$.
3. Calculate $\phi(n) = (p - 1)(q - 1) = 16 \times 10 = 160$.
4. Select e such that e is relatively prime to $\phi(n) = 160$ and less than $\phi(n)$; we choose $e = 7$.
5. Determine d such that $de \equiv 1 \pmod{160}$ and $d < 160$. The correct value is $d = 23$, because $23 \times 7 = 161 = (1 \times 160) + 1$; d can be calculated using the extended Euclid's algorithm (Chapter 2).

The resulting keys are public key $PU = \{7, 187\}$ and private key $PR = \{23, 187\}$. The example shows the use of these keys for a plaintext input of $M = 88$. For encryption, we need to calculate $C = 88^7 \bmod 187$. Exploiting the properties of modular arithmetic, we can do this as follows.

$$\begin{aligned}
88^7 \bmod 187 &= [(88^4 \bmod 187) \times (88^2 \bmod 187) \\
&\quad \times (88^1 \bmod 187)] \bmod 187 \\
88^1 \bmod 187 &= 88 \\
88^2 \bmod 187 &= 7744 \bmod 187 = 77 \\
88^4 \bmod 187 &= 59{,}969{,}536 \bmod 187 = 132 \\
88^7 \bmod 187 &= (88 \times 77 \times 132) \bmod 187 = 894{,}432 \bmod 187 = 11
\end{aligned}$$

For decryption, we calculate $M = 11^{23} \bmod 187$:

$$\begin{aligned}
11^{23} \bmod 187 &= [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \\
&\quad \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187 \\
11^1 \bmod 187 &= 11 \\
11^2 \bmod 187 &= 121 \\
11^4 \bmod 187 &= 14{,}641 \bmod 187 = 55 \\
11^8 \bmod 187 &= 214{,}358{,}881 \bmod 187 = 33 \\
11^{23} \bmod 187 &= (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 \\
&= 79{,}720{,}245 \bmod 187 = 88
\end{aligned}$$

5、形式化表达

Key Generation by Alice	
Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calcuate $\phi(n) = (p - 1)(q - 1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Public Key	
Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

6、回到举例 2：

823912106125050294393 7=2547896521897 * 3233695321

$$\psi(n) = \psi(823912106125050294393 7)$$

$$\psi(n) = \psi(p) \cdot \psi(q) = (p-1)(q-1) = (2547896521897-1)(3233695321-1)$$

公钥为 e，私钥为 d，$\gcd(e, \psi(n)) = 1$、$e \cdot d \bmod \psi(n) = 1$

$$c \leftarrow m^e \bmod n$$

$$m \leftarrow c^d \bmod n$$

$$m^{e \cdot d \bmod \psi(n)} \bmod n = m^{i\psi(n)+1} \bmod n = m$$

举例：已知 q=11，p=13，n=11*13=143

$$\psi(143) = (11-1)(13-1) = 120$$

选择私钥 $sk = 17$；计算公钥：$17 \cdot pk - i \cdot 120 = 1$ 二元一次方程能够在多项式时间内求解

$$17 \cdot pk = i \cdot 120 + 1$$

```
for (i=1, i<=119, i++) do
    if (17* i mod120=1)
```

break;
out PK=i;

得到公钥 $pk=113$

任意消息 m=1,2,3,4,…,142，mod143
数字签名：(m^17 mod143)^113 mod143=m
签名：sigma :=(m^17 mod143)
校验：sigma^113 mod143 = = m

公钥加密：(m^113 mod143)^17 mod143=m
加密：C := (m^113 mod143)
解密：m := C^17 mod143

RSA 算法改进版

$$M=m\|r$$
$$\sigma \leftarrow Sig(SK,M)$$
$$Valid\ /\ Invalid \leftarrow Ver(PK,M,\sigma)$$

$RSA:(m,\sigma,pk)$
$RSA':(m,r,\sigma,pk)$ **安全性更高。**

3 群

符号说明：a^b 是指 a 的 b 次方；a_1 下划线表示下标；a*b 是指 a 乘以 b。

介绍密码算法之前先要介绍一个关键的数学工具：**群**。
群定义：一个**集合 G**，满足以下 6 个条件，则称为群。
1.非空集：集合中至少有一个元素。
2.二元运算：集合中的元素能够进行一种运算，例如**加法**运算、或**乘法**运算。
3.封闭性：集合中的元素进行运算后，得到的结果仍然是**集合中**的元素。
4.结合律：任意 a,b,c 属于 G，则(a+b)+c=a+(b+c)。
5.单位元 e：加法情况下 a+e=e+a=a，乘法情况下：a*e=e*a=a。
6.每个元素都有逆元 a^(-1)：加法情况下 a+ a^(-1)= a^(-1)+a=e，乘法情况下：a*a^(-1)= a^(-1)*a=e。
因此集合 G 称为**群**。
对群的概念可以**简单理解**为：**具有封闭运算的集合称为群。**

举例：{0,1}集合，除法，则不满足封闭性。1 除以 0 等于无穷大。

一共有 6 个性质，主要用到二元运算、封闭性、单位元、**逆元**这四个性质。
而非空集和结合律很容易满足。
但是，i,j,k 虚数、向量、矩阵不满足结合律，不能用于构造**群**。

定义 1：如果一个群元素能够通过有限次**本身运算**，表达出群内其他所有元素，则称为群的**生成元**。**生成元：**用一个元素表达所有元素；
定义 2：群内元素**个数**称为群的**阶**。

例 1：集合{0,1,2,3,4,5,6}模系数为 7，就是一个**加法群**。
1.非空集：群内有 7 个元素。
2.二元运算:加法。
3.封闭性：群内任意两个元素相加后模 7 后仍然是群中的元素，例如(5+6)mod7=4；
4.结合律：((3+4)+5) mod7=(3+(4+5)) mod7=5，结果相同。
5.单位元为 e=0：3+0=0+3=3。
6.逆元：1 的逆元为 6，因为(1+6)mod7=0，2 的逆元为 5，因为(2+5)mod7=e，同理 3 的逆元为 4。0 的逆元是 0。
因此集合{0,1,2,3,4,5,6}模系数为 7 就是一个**群**。

7 是素数，这个素数群的性质特别好。因为素数 7 与群元素 i 是互素的，所以每个非零元素都是群的*生成元*。例如：群元素 2 能够通过有限次本身运算，表达其他所有元素：
(2+2)mod7=4 则表达群元素 4，
(2+2+2)mod7=6 则表达群元素 6，
(2+2+2+2)mod7=1 则表达群元素 1，
(2+2+2+2+2)mod7=3 则表达群元素 3，
(2+2+2+2+2+2)mod7=5 则表达群元素 5，
(2+2+2+2+2+2+2)mod7=0 则表达群元素 0。

群元素 3 能够通过有限次运算，表达其他所有元素：
(3)mod7=3
(3+3)mod7=6
(3+3+3)mod7=2
(3+3+3+3)mod7=5
(3+3+3+3+3)mod7=1
(3+3+3+3+3+3)mod7=4
(3+3+3+3+3+3+3)mod7=0

群元素 4 能够通过有限次运算，表达其他所有元素：
(4)mod7=4
(4+4)mod7=1
(4+4+4)mod7=5
(4+4+4+4)mod7=2
(4+4+4+4+4)mod7=6
(4+4+4+4+4+4)mod7=3
(4+4+4+4+4+4+4)mod7=0

群元素 5 能够通过有限次运算，表达其他所有元素：
(5)mod7=5
(5+5)mod7=3
(5+5+5)mod7=1
(5+5+5+5)mod7=6
(5+5+5+5+5)mod7=4
(5+5+5+5+5+5)mod7=2
(5+5+5+5+5+5+5)mod7=0

群元素 6 能够通过有限次运算，表达其他所有元素：
(6)mod7=6
(6+6)mod7=5
(6+6+6)mod7=4
(6+6+6+6)mod7=3
(6+6+6+6+6)mod7=2
(6+6+6+6+6+6)mod7=1
(6+6+6+6+6+6+6)mod7=0

群元素 1,2,3,4,5,6 均可以通过有限次运算表达其他群元素。

例 2：集合{1,2,3,4,5,6}模系数为 7，就是一个**乘法群**。
1.非空集：群内有 6 个元素。
2.二元运算:乘法。
3.封闭性：群内任意两个元素相乘后模 7 后仍然是群中的元素，例如(5*6)mod7=2；
4.结合律：((3*4) *5) mod7=(3* (4*5)) mod7=4，结果相同。
5.单位元为 1：1 乘以任意元素等于任意元素; 3*1=1*3=3。
6.逆元：1 的逆元为 1，因为(1*1)mod7=1; 2 的逆元为 4，因为(2*4)mod7=e=1; 3 的逆元为 5，因为(3*5)mod7=1。6 的逆元为 6，因为(6*6)mod7=1。
因此，集合{1,2,3,4,5,6}模系数为 7 就是一个**乘法群**。

(2)mod7=2
(2*2)mod7=4
(2*2*2)mod7=1
(2*2*2*2)mod7=2
(2*2*2*2*2)mod7=4
(2*2*2*2*2*2)mod7=1
(2*2*2*2*2*2*2)mod7=2
只能表达 1,2,4 因此 2 不是生成元

(3)mod7=3，记为 $3^1 \bmod 7 = 3$

(3*3)mod7=2，记为 $3^2 \bmod 7 = 2$

(3*3*3)mod7=6，记为 $3^3 \bmod 7 = 6$

(3*3*3*3)mod7=4，记为 $3^4 \bmod 7 = 4$

(3*3*3*3*3)mod7=5，记为 $3^5 \bmod 7 = 5$

(3*3*3*3*3*3)mod7=1，记为 $3^6 \bmod 7 = 1$

所以 3 是生成元

举例：公钥为 6，生成元是 3；如何计算私钥？需要 for 循环，指数时间

- **已知私钥，能够快速计算公钥；**
- **已知公钥，需要指数时间计算私钥。**

已知公钥计算私钥，需要暴力搜索，短时间内不可行，需要指数时间。
已知私钥，计算公钥，多项式时间内可计算；

离散对数困难问题：

私钥 $sk = x$，公钥 $pk = X$，其中 $\begin{matrix} X = g^x \\ PK = g^{sk} \end{matrix}$。已知 g, X，不能在多项式时间内计算出 x。

(3*3*3*3*3)mod7=5，记为 $3^{5=sk} \bmod 7 = 5 = PK$

4Diffie-Hellman 密钥交换

Alice 私钥 $SK_1 = \alpha$，公钥 $PK_1 = g^{\alpha}$；

Bob 私钥 $SK_2 = \beta$，公钥 $PK_2 = g^{\beta}$；

Diffie-Hellman 交互协议：

- Alice 发送其公钥 PK_1 给 Bob；
- Bob 发送其公钥 PK_2 给 Alice。

则 Alice 如下计算 $(PK_2)^{SK_1} = (g^{\beta})^{\alpha} = g^{\alpha\beta}$；Bob 如下计算 $(PK_1)^{SK_2} = (g^{\alpha})^{\beta} = g^{\alpha\beta}$

公共密钥、双方的保密随机数

因此 Alice 与 Bob 计算出相同的**公共密钥** $Key = g^{\alpha\beta}$，或 $Key = Hash\left(g^{\alpha\beta}\right)$

A	p, g	**B**

$SK_1 = \alpha$，$PK_1 = g^{\alpha}$	$\xrightarrow{PK_1}$	
	$\xleftarrow{PK_2}$	$SK_2 = \beta$，$PK_2 = g^{\beta}$
$Key = (PK_2)^{\alpha}$	$g^{\alpha\beta}$	$Key = (PK_1)^{\beta}$

举例：乘法素数群 $n=11$，生成元 $g=2$，乘法群。

Alice 私钥 $SK_1=\alpha=4$，公钥 $PK_1=g^{\alpha}=2^4 \bmod 11=5$；

Bob 私钥 $SK_2=\beta=5$，公钥 $PK_2=g^{\beta}=2^5 \bmod 11=10$；

协议：Alice 发送其公钥 $PK_1=5$ 给 Bob；Bob 发送其公钥 $PK_2=10$ 给 Alice。

则 Alice 如下计算 $(PK_2)^{SK_1}=(10)^4 \bmod 11=1$；Bob 如下计算 $(PK_1)^{SK_2}=(5)^5 \bmod 11=1$

等价于 Alice，计算 $Key=(2^5)^4 \bmod 11=1$，Bob 计算 $Key=(2^4)^5 \bmod 11=1$

因此 Alice 与 Bob 计算出相同的**会话密钥** $Key=g^{\alpha\beta}=2^{4*5}=1$，

或 $Key=Hash\left(g^{\alpha\beta}\right)=SHA256(2^{4*5} \bmod 11)=\mathrm{SHA}\,256(1)$

但是，有 2 个缺点：
缺点 1：公共密钥永远没变化
改进：添加公开随机数 r
优化协议：

Alice 发送其公钥 PK_1 和公开随机数 r_1 给 Bob；

Bob 发送其公钥 PK_2 和公开随机数 r_2 给 Alice

Alice 如下计算 $(PK_2)^{SK_1}=(g^{\beta})^{\alpha}=g^{\alpha\beta}$，令 $Key=Hash\left(g^{\alpha\beta},r_1,r_2\right)$

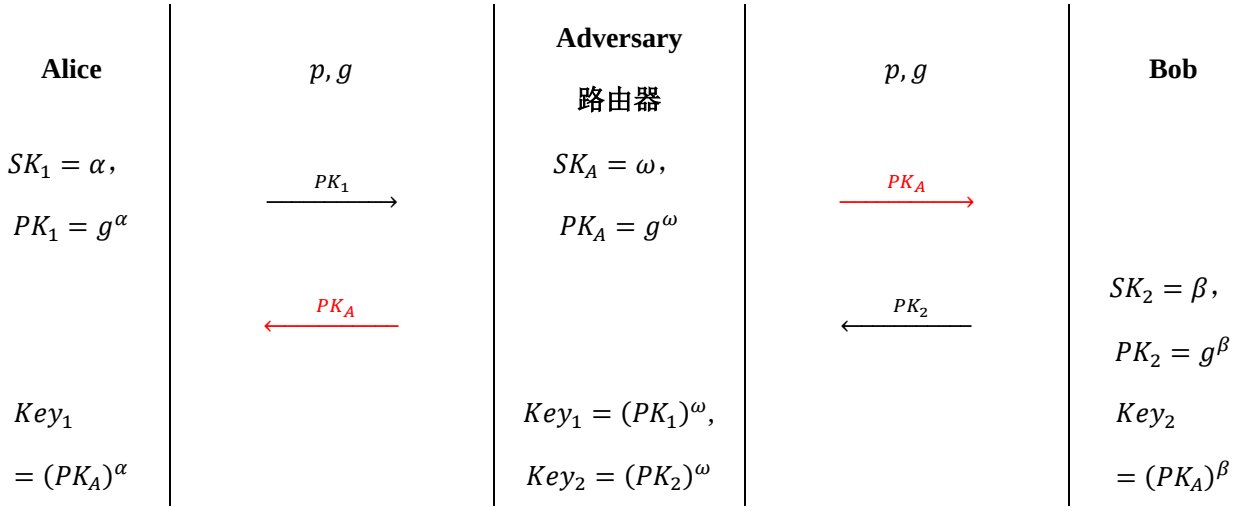
Bob 如下计算 $(PK_1)^{SK_2}=(g^{\alpha})^{\beta}=g^{\alpha\beta}$，令 $Key=Hash\left(g^{\alpha\beta},r_1,r_2\right)$

因此 Alice 与 Bob 计算出相同的会话密钥 Key。会话密钥每次都会发生变化！

缺点 2：中间人攻击

Adversary 私钥 $SK_A=\omega$，公钥 $PK_A=g^{\omega}$

理想情况：Alice 发送其公钥 PK_1 给 Bob；Bob 发送其公钥 PK_2 给 Alice；



实际情况：

Alice 发送其公钥 PK_1 给 Adversary，Adversary 发送其公钥 PK_A 给 Bob。

Bob 发送其公钥 PK_2 给 Adversary，Adversary 发送其公钥 PK_A 给 Alice

则 Alice 计算出与 Adversary 的会话密钥 $Key_1 = (PK_A)^{SK_1} = (g^{\omega})^{\alpha} = g^{\alpha \cdot \omega}$

Adversary 计算出与 Alice 的会话密钥 $Key_1 = (PK_1)^{SK_A} = (g^{\alpha})^{\omega} = g^{\alpha \cdot \omega}$

则 Bob 计算出与 Adversary 的会话密钥 $Key_2 = (PK_A)^{SK_2} = (g^{\omega})^{\beta} = g^{\beta \cdot \omega}$

Adversary 计算出与 Bob 的会话密钥 $Key_2 = (PK_2)^{SK_A} = (g^{\beta})^{\omega} = g^{\beta \cdot \omega}$

添加随机数不能解决中间人攻击

理想情况：

Alice 发送公钥 PK_1 和随机数 r_1 给 Bob；Bob 发送公钥 PK_2 和随机数 r_2 给 Alice

实际情况：

Alice 发送公钥 PK_1 和随机数 r_1 给 Adversary，Adversary 发送公钥 PK_A 和随机数 r_1 给 Bob

Bob 发送公钥 PK_2 和随机数 r_2 给 Adversary，Adversary 发送公钥 PK_A 和随机数 r_2 给 Alice

（1）Alice 计算与 Adversary 的会话密钥

$$Key_1 = Hash\left((PK_A)^{SK_1}, r_1, r_2\right) = Hash\left(g^{\alpha \cdot \omega}, r_1, r_2\right)$$

（1）Adversary 计算出与 Alice 的会话密钥

$$Key_1 = Hash\left((PK_1)^{SK_A}, r_1, r_2\right) = Hash\left(g^{\alpha \cdot \omega}, r_1, r_2\right)$$

（2）Bob 计算与 Adversary 的会话密钥

$$Key_2 = Hash\left((PK_A)^{SK_2}, r_1, r_2\right) = Hash\left(g^{\beta\cdot\omega}, r_1, r_2\right)$$

（2）Adversary 计算与 Bob 的会话密钥

$$Key_2 = Hash\left((PK_2)^{SK_A}, r_1, r_2\right) = Hash\left(g^{\beta\cdot\omega}, r_1, r_2\right)$$

需要一个**认证**，确保接收到的公钥是对方的，而不是 Adversary 的。

发公钥证书才能够解决中间人攻击，攻击者只能截断，不能窃听，但是证书是中心化的系统有拒绝服务攻击和反应迟缓等问题。

三方 Diffie-Hellman 密钥交换协议，类比到多方

Alice 私钥 $SK_1 = \alpha$，公钥 $PK_1 = g^{\alpha}$；

Bob 私钥 $SK_2 = \beta$，公钥 $PK_2 = g^{\beta}$；

Carol 私钥 $SK_3 = \varepsilon$，公钥 $PK_3 = g^{\varepsilon}$

协议 1： Alice 发送其公钥 PK_1 给 Bob；Bob 发送其公钥 PK_2 给 Alice。

则 Alice 如下计算 $(PK_2)^{SK_1} = (g^{\beta})^{\alpha} = g^{\alpha\beta}$；Bob 如下计算 $(PK_1)^{SK_2} = (g^{\alpha})^{\beta} = g^{\alpha\beta}$

因此 Alice 与 Bob 计算出相同的**会话密钥** $Key_{\pi} = r = Hash\left(g^{\alpha\beta}\right)$，对应公钥 $PK_{\pi} = g^{R}$

协议 2： Alice 和 Bob 将 PK_{π} 发送给 Carol，Carol 将 PK_3 发送给 Alice 和 Bob。

Alice/Bob 计算：$Key_{\psi} = Hash\left((PK_3)^{r}\right) = Hash(g^{r\cdot\varepsilon})$；

Carol 校验 Alice 和 Bob 发送过来的 PK_{π} 相等， Carol 计算：

$$Key_{\psi} = Hash\left((PK_{\pi})^{SK_3}\right) = Hash(g^{r\cdot\varepsilon})$$

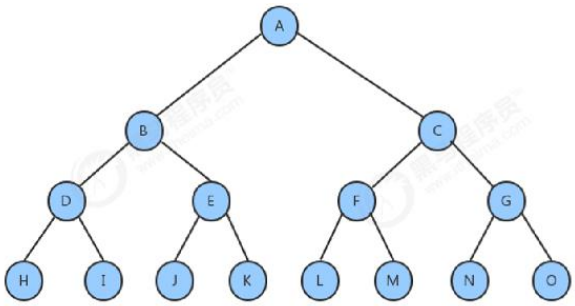
所以 Alice, Bob，Carol 计算出共同的三方公共密钥 Key_{ψ}。

再加一个参与方，则 Alice, Bob，Carol 计算共同的会话密钥 Key_{ψ} 对于的公钥

$PK_{\psi} = g^{Key_{\psi}}$

四个参与方一起计算共享会话密钥，直到 n 个参与方共享会话密钥。用共享的会话密钥和 AES 加密算法加密数据。

缺点：每进来一个参与方，就要协商一次，复杂度呈线性增加。
优化方案：两两协商，形成一个二叉树，协商次数是树的高度。



有些情况下，只能不断增加**参与方**，只能线性增加。

5ElGamal 加密

SysGen: The system parameter generation algorithm takes as input a security parameter λ. It chooses a cyclic group $(\mathbb{G}, p, g)$ and returns the system parameters $SP = (\mathbb{G}, p, g)$.

KeyGen: The key generation algorithm takes as input the system parameters SP. It randomly chooses $\alpha \in \mathbb{Z}_p$, computes $g_1 = g^{\alpha}$, and returns a public/secret key pair (pk, sk) as follows:

$$pk = g_1, \quad sk = \alpha.$$

Encrypt: The encryption algorithm takes as input a message $m \in \mathbb{G}$, the public key pk, and the system parameters SP. It chooses a random number $r \in \mathbb{Z}_p$ and returns the ciphertext CT as

$$CT = (C_1, C_2) = \left(g^r, \ g_1^r \cdot m\right).$$

Decrypt: The decryption algorithm takes as input a ciphertext CT, the secret key sk, and the system parameters SP. Let $CT = (C_1, C_2)$. It decrypts the message by computing

$$C_2 \cdot C_1^{-\alpha} = g_1^r m \cdot (g^r)^{-\alpha} = m.$$

$$(g^{\alpha})^{-r} = g_1^{-r}$$

对称双线性映射

The definition of symmetric pairing is stated as follows. Let $\mathbb{PG} = (\mathbb{G}, \mathbb{G}_T, g, p, e)$ be a symmetric-pairing group. Here, $\mathbb{G}$ is an elliptic curve subgroup, $\mathbb{G}_T$ is a multiplicative subgroup, $|\mathbb{G}| = |\mathbb{G}_T| = p$, g is a generator of $\mathbb{G}$, and e is a map satisfying the following three properties.

- For all $u, v \in \mathbb{G}, a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$.
- $e(g, g)$ is a generator of group $\mathbb{G}_T$.
- For all $u, v \in \mathbb{G}$, there exist efficient algorithms to compute $e(u, v)$.

非对称双线性映射

The definition of asymmetric pairing (Asymmetric 2) is stated as follows. Let $\mathbb{PG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, p, e)$ be an asymmetric-pairing group. Here, $\mathbb{G}_1, \mathbb{G}_2$ are elliptic curve subgroups, $\mathbb{G}_T$ is a multiplicative subgroup, $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| = p$, g_1 is a generator of $\mathbb{G}_1$, g_2 is a generator of $\mathbb{G}_2$, and e is a map satisfying the following three properties.

- For all $u \in \mathbb{G}_1, v \in \mathbb{G}_2, a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$.
- $e(g_1, g_2)$ is a generator of group $\mathbb{G}_T$.
- For all $u \in \mathbb{G}_1, v \in \mathbb{G}_2$, there exist efficient algorithms to compute $e(u, v)$.

映射先计算与后计算，无所谓。

6BLS 签名方案

特殊的哈希函数：把 msg 映射到椭圆曲线点上

哈希函数：3 类：

- **情况 1：映射到群 G 上，复杂度最高；**
- **SHA3 产生随机数**
- **SHA3modp 映射到 Z_p**

SysGen: The system parameter generation algorithm takes as input a security parameter λ. It chooses a pairing group $\mathbb{PG} = (\mathbb{G}, \mathbb{G}_T, g, p, e)$, selects a cryptographic hash function $H : \{0,1\}^* \to \mathbb{G}$, and returns the system parameters $SP = (\mathbb{PG}, H)$.

KeyGen: The key generation algorithm takes as input the system parameters SP. It randomly chooses $\alpha \in \mathbb{Z}_p$, computes $h = g^\alpha$, and returns a public/secret key pair (pk, sk) as follows:

$$pk = h, \quad sk = \alpha.$$

Sign: The signing algorithm takes as input a message $m \in \{0,1\}^*$, the secret key sk, and the system parameters SP. It returns the signature σ_m on m as

$$\sigma_m = H(m)^\alpha.$$

Verify: The verification algorithm takes as input a message-signature pair (m, σ_m), the public key pk, and the system parameters SP. It accepts the signature if

$$e(\sigma_m, g) = e\big(H(m), h\big).$$

$$e(\sigma_m, g) = e(H(m)^{\alpha}, g) = e(H(m), g)^{\alpha} = e(H(m), g^{\alpha}) = e(H(m), h)$$

映射先计算与后计算，无所谓。

BLS 签名仅有 1 个随机因子：私钥

扩展：令 $M' = \{M \mid r\}$

签名：$\sigma = H(M')^{\alpha}$

广播：(M, r, σ, pk)

任意验证方：

（1）数据拼接：$M' \leftarrow \{M \mid r\}$

（2）校验：$e(\sigma, g) = e(H(M'), h)$

- **BLS 原方案，安全性损失 30-40bit；安全性 48*8-30**
- **如果 r 是 1bit 的随机数，则安全性损失仅 1bit；安全性 48*8-1**
- **如果 r 是随机数，与私钥位宽一样，则安全性无损失；私钥 48*8bit，则安全程度就是 48*8bit**

3 聚合签名

上述 BLS 签名方案具有以下三个关键性质：1 允许把不同实体对不同消息的签名集合到一个签名中，且拥有所有签名的任意节点均能够聚集签名。2 允许加入新成员。3 任意两个签名集合后，允许第三个签名聚合到该签名集合中。

对任意用户 i，其中 $i = 1, ..., n$，私钥为 $x_i \in \mathbb{Z}_p$，公钥为 $v_i = g_2^{x_i} \in G_2$。

- **聚合签名**：用户 i 对一个消息 $M_i \in \{0,1\}^*$ 签名获得 $\sigma_i = H(M_i)^{x_i} \in G_1$。聚集所有的签名，则仅需要如下计算

$$\sigma_{1,2,...,n} \leftarrow \sigma_1 \cdot \sigma_2 \cdot ... \cdot \sigma_n \in G_1.$$

- **验证**：给定对应的所有公钥 $v_1, .., v_n \in G_2$，所有的消息 $M_1, ..., M_n \in \{0,1\}^*$，聚集签名为 $\sigma_{1,2,...,n} \in G_1$。验证对于所有的用户 i 签名的消息 M_i，$i = 1, ..., n$，如下检测：
  1. 消息 $M_1, ..., M_n$ 互不相同；
  2. $e(\sigma_{1,2,...,n}, g_2) = \prod_{i=1}^{n} e(H(M_i), v_i)$。

  如果上述两个条件均成立，则接受聚集签名，否则拒绝。

公式推导：

$$e(u^x \cdot u^y, v) = e(u^{x+y}, v) = e(u, v)^{x+y} = e(u, v)^x \cdot e(u, v)^y = e(u^x, v) \cdot e(u^y, v)$$

$$let: a = u^x, b = u^y$$

$$then:$$

$$e(u^x \cdot u^y, v) = e(a \cdot b, v)$$

$$e(u^x, v) \cdot e(u^y, v) = e(a, v) \cdot e(b, v)$$

$$therefore:$$

$$e(a \cdot b, v) = e(a, v) \cdot e(b, v)$$

验证：

$$e(\sigma_{1,2,...,n}, g_2) = e(\sigma_1 \cdot \sigma_2 \cdot ... \cdot \sigma_n, g_2) = e(\sigma_1, g_2) \cdot e(\sigma_2, g_2) \cdot ... \cdot e(\sigma_n, g_2)$$

$$= \prod_{i=1}^{n} e(\sigma_i, g_2) = \prod_{i=1}^{n} e(H(M_i)^{x_i}, g_2) = \prod_{i=1}^{n} e(H(M_i), g_2)^{x_i} = \prod_{i=1}^{n} e(H(M_i), g_2^{x_i})$$

$$= \prod_{i=1}^{n} e(H(M_i), v_i)$$

4 批验证

假设 n 个用户对同一个消息 M 签名，则能够实现批验证，且验证速度极快。任意用户获得的 n 个签名为 $\sigma_1,...,\sigma_n$，则能够对这 n 个签名进行批验证，其验证速度远远快于逐个验证。

用户 i 的私钥 $x_i \in \mathbb{Z}_p$，公钥 $v_i \in g_2^{x_i} \in G_2$，签名 $\sigma_i = H(M)^{x_i} \in G_1$，则

1. 随机选择 n 个整数 $c_1,...,c_n \in [0,B]$，B 为某个固定值；
2. 计算 $V \leftarrow \prod_{i=1}^{n} v_i^{c_i} \in G_2, U \leftarrow \prod_{i=1}^{n} \sigma_i^{c_i} \in G_1$；
3. 如下检测等式是否成立，$e(U,g_2) = e(H(M),V)$。

如果满足上述两个条件，则接受所有的 n 个签名，否则拒绝。

$$e\left(\prod_{i=1}^{n} a_i, b\right) = e(a_1 \cdot a_2 \cdot ... \cdot a_n, b) = e(a_1,b) \cdot e(a_2,b) \cdot ... \cdot e(a_n,b) = \prod_{i=1}^{n} e(a_i,b)$$

$$U \leftarrow \prod_{i=1}^{n} \sigma_i^{c_i} = \prod_{i=1}^{n} H(M)^{x_i \cdot c_i}, V \leftarrow \prod_{i=1}^{n} v_i^{c_i} = \prod_{i=1}^{n} g_2^{x_i \cdot c_i}$$

单个

$$e(U,g_2) = e\left(\prod_{i=1}^{n} H(M)^{x_i}, g_2\right) = \prod_{i=1}^{n} e\left(H(M)^{x_i}, g_2\right) = \prod_{i=1}^{n} e\left(H(M), g_2\right)^{x_i}$$

$$e(H(M),V) = e\left(H(M), \prod_{i=1}^{n} g_2^{x_i}\right) = \prod_{i=1}^{n} e\left(H(M), g_2^{x_i}\right) = \prod_{i=1}^{n} e\left(H(M), g_2\right)^{x_i}$$

多个

$$e(U,g_2) = e\left(\prod_{i=1}^{n} H(M)^{x_i \cdot c_i}, g_2\right) = \prod_{i=1}^{n} e\left(H(M)^{x_i \cdot c_i}, g_2\right) = \prod_{i=1}^{n} e\left(H(M), g_2\right)^{x_i \cdot c_i}$$

$$e(H(M),V) = e\left(H(M), \prod_{i=1}^{n} g_2^{x_i \cdot c_i}\right) = \prod_{i=1}^{n} e\left(H(M), g_2^{x_i \cdot c_i}\right) = \prod_{i=1}^{n} e\left(H(M), g_2\right)^{x_i \cdot c_i}$$

双线性映射：计算复杂度很高；尽量少算；
其他群运算计算复杂度很低，可以多算。
N 签名累乘　映射　n 个公钥累乘（牛头对马嘴的概率很高。

Alice 对 m1 签名 sigma1；Bob 对 m2 签名 sigma2；
牛头对马嘴：Alice 对 m2 签名 sigma1，Bob 对 m1 签名 sigma2；
两种情况计算出来的双线性映射相同；但是，添加随机数后，两种情况计算出来的结果是不同的，碰撞概率是降低的。B 等于 2^40 次方，发生碰撞的概率可忽略。

7 零知识证明

W Zero-knowledge proof - W

Abstract examples [edit]

The Ali Baba cave [edit]

There is a well-known story presenting the fundamental ideas of zero-knowledge proofs, first published by Jean-Jacques Quisquater and others in their paper "How to Explain Zero-Knowledge Protocols to Your Children".[4] It is common practice to label the two parties in a zero-knowledge proof as Peggy (the **prover** of the statement) and Victor (the **verifier** of the statement).

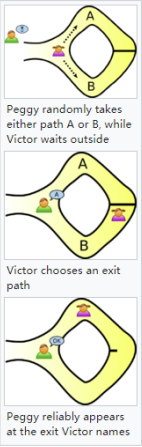
In this story, Peggy has uncovered the secret word used to open a magic door in a cave. The cave is shaped like a ring, with the entrance on one side and the magic door blocking the opposite side. Victor wants to know whether Peggy knows the secret word; but Peggy, being a very private person, does not want to reveal her knowledge (the secret word) to Victor or to reveal the fact of her knowledge to the world in general.

They label the left and right paths from the entrance A and B. First, Victor waits outside the cave as Peggy goes in. Peggy takes either path A or B; Victor is not allowed to see which path she takes. Then, Victor enters the cave and shouts the name of the path he wants her to use to return, either A or B, chosen at random. Providing she really does know the magic word, this is easy: she opens the door, if necessary, and returns along the desired path.

However, suppose she did not know the word. Then, she would only be able to return by the named path if Victor were to give the name of the same path by which she had entered. Since Victor would choose A or B at random, she would have a 50% chance of guessing correctly. If they were to repeat this trick many times, say 20 times in a row, her chance of successfully anticipating all of Victor's requests would become vanishingly small (about one in a million).

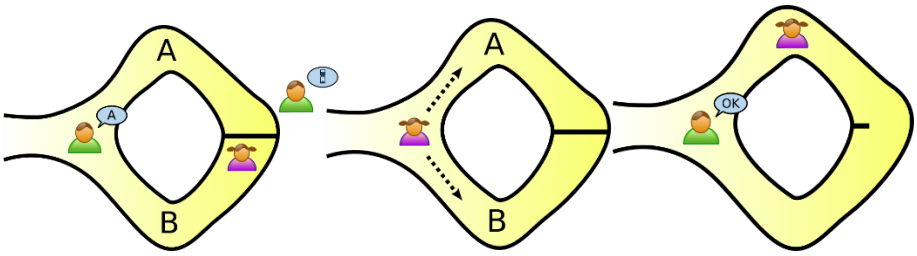
Thus, if Peggy repeatedly appears at the exit Victor names, he can conclude that it is extremely probable that Peggy does in fact know the secret word.

One side note with respect to third-party observers: even if Victor is wearing a hidden camera that records the whole transaction, the only thing the camera will record is in one case Victor shouting "A!" and Peggy appearing at A or in the other case Victor shouting "B!" and Peggy appearing at B. A recording of this type would be trivial for any two people to fake (requiring only that Peggy and Victor agree



简介非交互式知识论证

把任意运算关系 构造为 NP 问题；电路



阿里巴巴 cave

运行 n 次；

三色图：解释零知识证明

小红每次都正确 1/2^n

概率

但是，如果存在 C 通道，则小明被坑了。

现有的密码初始化设置不能有后门；拥有后门 可以诈骗。

群定义：一个**集合 G**，满足以下 6 个条件，则称为群。

1.非空集：集合中至少有一个元素。

2.二元运算：集合中的元素能够进行一种运算，例如**加法**运算、或**乘法**运算。

3.封闭性：集合中的元素进行运算后，得到的结果仍然是集合中的元素。
4.结合律：任意 a,b,c 属于 G，则(a+b)+c=a+(b+c)。
5.单位元 e：加法情况下 a+e=e+a=a，乘法情况下：a*e=e*a=a。

6.每个元素都有逆元 a^{-1}：加法情况下 $a+a^{-1}=a^{-1}+a=e$，乘法情况下：

$a\cdot a^{-1}=a^{-1}\cdot a=e$。

因此集合 G 称为群。
对群的概念可以简单理解为：**具有封闭运算的集合称为群。**

只有离散对数是困难的，计算逆元、单位元等等都可以在多项式时间内完成，

所以可以对公式进行各种变形。

例 1：集合{1,2,3,4,5,6}模系数为 7，就是一个**乘法群**。

(3)mod7=3，记为 $3^1 \bmod 7=3$

(3*3)mod7=2，记为 $3^2 \bmod 7=2$

(3*3*3)mod7=6，记为 $3^3 \bmod 7=6$

(3*3*3*3)mod7=4，记为 $3^4 \bmod 7=4$

(3*3*3*3*3)mod7=5，记为 $3^5 \bmod 7=5$

(3*3*3*3*3*3)mod7=1，记为 $3^6 \bmod 7=1$

所以 3 是生成元
2=2
2*2=4
2*2*2=1
2*2*2*2=2
2*2*2*2*2=4

离散对数困难问题：

私钥 $sk=x$，公钥 $pk=X$，其中 $\begin{matrix}X=g^x\\PK=g^{sk} \bmod p\end{matrix}$。已知 g,X，不能在多项式时间内计算出 x。

```
for (i=1, i<=p, i++){
    if ( PK = g^i mod p )
    output i.
    break.
```

} // $p = 2^{256} + 11$

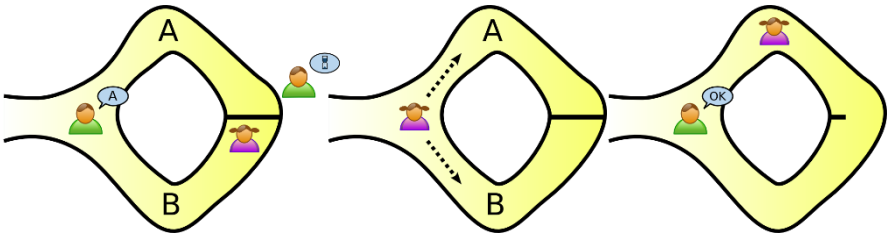
只有离散对数是困难的，计算逆元、单位元等等都可以在多项式时间内完成。

Alice 向 Bob 证明他知道某个公钥 PK 对应的私钥 SK： $PK = g^{sk} \bmod p$

公钥加密：Bob 用 Alice 公钥加密消息并发送，Alice 接收方能解密。
承诺：

数字签名：用 Alice 私钥签名，验证方用 Alice 公钥验证。
零知识证明
A zero-knowledge proof must satisfy three properties:

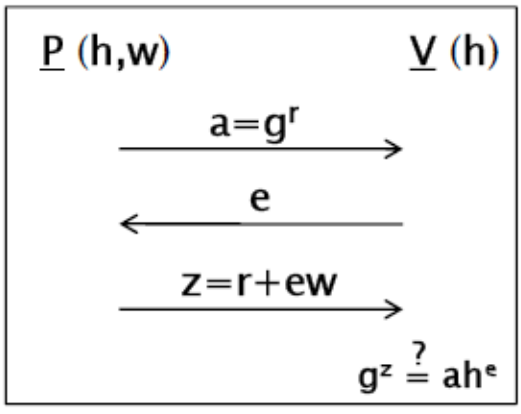
- **Completeness:** 诚实证明方能够让诚实验证方接受结果。
- **Soundness:** 恶意证明方不能让诚实验证方接受一个错误的结果。SK1=SK2
- **Zero-knowledge:** 诚实验证方仅能接受结果而不知道任意额外的信息。



Sigma 协议

系统参数：Let G be a group of order q, with generator g；

P and **V** have input $h \in G$, P has ω such that $h = g^{\omega}$；



1（承诺）P 选择随机数 $r \bmod q$，计算 $a = g^r$，发送 a；

2（挑战）V 选择随机数 e，发送 e；

3（响应）P 计算 $z = r + e \cdot \omega \bmod q$，发送 **z**；*注意：在群的阶范围内构造了线性关系*

4（验证）V 验证，如果等式 $g^z == a \cdot h^e$ 成立，则接受，否则拒绝。

$$g^z = g^{r+e\cdot\omega} = g^r \cdot (g^\omega)^e = a \cdot h^e$$

在群上验证线性关系，但是不知道 r 和 ω

如果证明方不知道 ω，则证明方输出的 z 不能使得等式成立。
因此，由于第 3 步 P 的构造必须使用 r, e, ω，否则公式不成立，V 不接受结果。

非交互式零知识证明

系统参数：Let G be a group of order q, with generator g；

P and **V** have input $h \in G$, P has ω such that $h = g^\omega$；

1（承诺）P 选择随机数 r，计算 $a = g^r$

2（挑战）P 计算随机数 $e = H(h, a)$

3（响应）P 计算 $z = r + e \cdot \omega$，将发送 a, z

4（验证）V 验证，计算 $e = H(h, a)$ 如果等式 $g^z == a \cdot h^e$ 成立，则接受，否则拒绝。

$$g^z = g^{r+e\cdot\omega} = g^r \cdot (g^\omega)^e = a \cdot h^e$$

$$\left\langle \begin{array}{r} g^z = a \cdot h^e \Rightarrow a = g^z \cdot h^{-e} \\ H(a) = e \end{array} \right\rangle \Rightarrow H(g^z \cdot h^{-e}) = e$$

额外条件：非交互式零知识证明要求哈希函数的输出是抗碰撞的!
如果哈希函数不能抵抗碰撞，则如下造假

1（承诺）P 选择随机数 r，计算 $a = g^r$

2（挑战）P 计算随机数 $e = H(h, a)$

3（响应）P 计算 $z = r + e \cdot \omega$，将发送 a, z

4（验证）V 验证，计算 $e = H(h, a)$ 如果等式 $g^z == a \cdot h^e$ 成立，则接受，否则拒绝。

For(i=0, , i++){

选择一个随机数 e 和 z，计算 $a := g^z h^{-e}$，然后计算 $e = H(h, a)$

If 如果哈希函抗碰撞，则 $e = H(h, a)$ 输出的随机数 e 与随机数 e 发生碰撞的概率可忽略。

If 如果哈希函不抗碰撞，则 $e = H(h,a) = H(*)$ 输出的随机数 e 与随机数 e 发生碰撞的概率不可忽略。

e==e,　break；

}

群定义：一个**集合 G**，满足以下 6 个条件，则称为群。

1.非空集：集合中至少有一个元素。

2.二元运算：集合中的元素能够进行一种运算，例如**加法**运算、或**乘法**运算。

3.封闭性：集合中的元素进行运算后，得到的结果仍然是集合中的元素。

4.结合律：任意 a,b,c 属于 G，则(a+b)+c=a+(b+c)。

5.单位元 e：加法情况下 a+e=e+a=a，乘法情况下：a*e=e*a=a。

6.每个元素都有逆元 a^(-1)：加法情况下 a+ a^(-1)= a^(-1)+a=e，乘法情况下：a*a^(-1)= a^(-1)*a=e。

因此集合 G 称为群。

对群的概念可以简单理解为：**具有封闭运算的集合称为群。**

只有离散对数是困难的，计算逆元、单位元等等都可以在多项式时间内完成，所以可以对公式进行各种变形。

其他都是形式化东西；群根本；

素数群；{0, 1, 2, 3, 4, 5, 6}

椭圆曲线点 P1, …, Pn（横坐标 纵坐标 对称关系 运算关系）椭圆曲线仅仅是一个轮廓；群才是本质；椭圆曲线群，也是群，用椭圆曲线上的离散坐标点表达群，而不是之前学的数字表达群。

8 椭圆曲线群

10.3.1　Abel 群

由第 4 章可知，Abel 群 G 由元素的集合及其上的二元运算 · 组成，有时记为 $\{G, \cdot\}$。将 G 中元素的序偶 (a,b) 与 G 中元素 $(a \cdot b)$ 对应，使得下述公理①成立：

(A1) 封闭性	若 a 和 b 属于 G，则 $a \cdot b$ 也属于 G。
(A2) 结合性	对 G 中任意的 a,b 和 c，$(a \cdot b) \cdot c = a \cdot (b \cdot c)$
(A3) 单位元	G 中存在元素 e，使得对 G 中所有的 a，$e \cdot a = a \cdot e = a$。
(A4) 逆元	对 G 中任何 a，存在 G 中元素 a'，使得 $a' \cdot a = a \cdot a' = e$。
(A5) 交换性	对 G 中任何 a 和 b，有 $a \cdot b = b \cdot a$。

椭圆曲线并不是椭圆，之所以称为椭圆曲线是因为它们与计算椭圆周长的方程相似，也是用三次方程来表示的。一般，椭圆曲线的三次方程形为

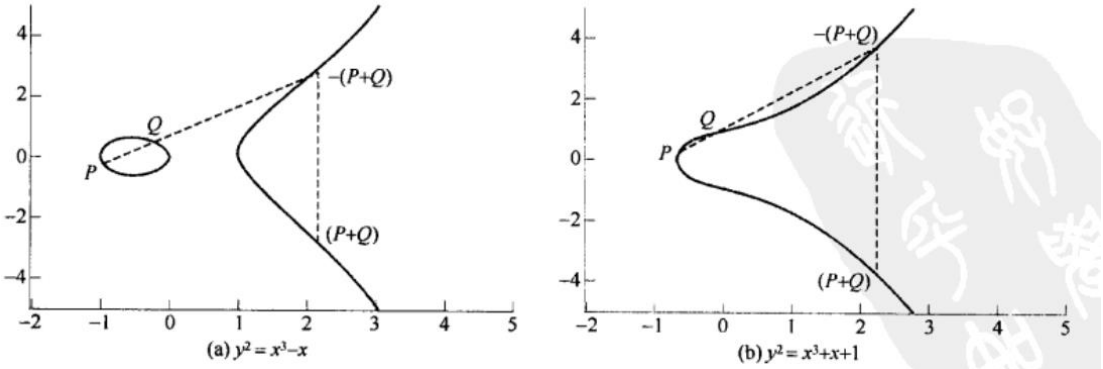
$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

其中 a,b,c,d 和 e 是实数，x 和 y 在实数集上取值①。对我们而言将方程限制为下述形式就已足够：

$$y^2 = x^3 + ax + b \qquad (10.1)$$

因为方程中的指数最高是3，所以我们称之为三次方程，或者说方程的次数为3。椭圆曲线的定义中还包含一个称为无穷远点或零点的元素，记为 O，我们以后再讨论这个概念。为了画出该曲线，我们需要计算：

$$y = \sqrt{x^3 + ax + b}$$



定义：点加运算为：P+Q=A

定义：倍点运算 k*P=A，k=2^n+…+2^0
Q=(2^n)*P
2P 在 P 点做切线，P1
2P1=2^2P 在 P1 点切线，P2
P2=2P1=2^3P
Pn=2^nP
k*P= Pn+…+P
最简单的假设：PK=2^n * G 椭圆曲线生成元；
有 2 个切点；A　B，他们横坐标 纵坐标都不等；4 切点
任意直线 与 曲线 相交 最多有 3 个解；
2^n 切点；PK=2^n * P
已知公钥 无法计算私钥，2^n 个点；

肯定只有 2 个解，规定第 3 个解是无穷远 或 0；
P+(-P)
无穷远或 0 就是群的：单位元
限制死了，P　　Q
椭圆曲线点：构成了一个**群**；
非空集合；
唯一运算关系：

交换律 结合律 每个元都有逆元 负元；

N=p*q 唯一分解；

RSA，是因为有大量的非素数，**合数不可用**，导致**亚指数，需要 1024 位。慢，数据量大。**

极度简单，非常安全。

A， -A 逆元，负元 加法群

A A^(-1)逆元；乘法群

P +(-P)= (x,y)+(x,-y)=(2x, 0)出现了一个 0，

单位元；

（0,0）（无穷远，无穷远）

（1）O 是加法的单位元。这样有 $O=-O$；对椭圆曲线上的任何一点 P，有 $P+O=P$。下面假定 $P\neq Q$ 且 $Q\neq O$。

（2）点 P 的负元是具有相同 x 坐标和相反的 y 坐标的点，即若 $P=(x,y)$，则 $-P=(x,-y)$。注意这两个点可用一条垂直的线连接起来，并且 $P+(-P)=P-P=O$。

（3）要计算 x 坐标不相同的两点 P 和 Q 之和，则在 P 和 Q 间作一条直线并找出第三个交点 R，显然存在有唯一的交点 R（除非这条直线在 P 或 Q 处与该椭圆曲线相切，此时我们分别取 $R=P$ 或 $R=Q$）。要形成群，需要定义如下三个点上的加法：$P+Q=-R$。也就是说，定义 $P+Q$ 为第三个交点（相对于 x 轴）的镜像。图 10.4 说明了这一情形。

（4）上述术语的几何解释也适用于具有相同 x 坐标的两个点 P 和 $-P$ 的情形。用一条垂直的线连接这两点，这也可视为在无穷远点处与曲线相交，因此有 $P+(-P)=O$，与上述步骤（2）相一致。

（5）为计算点 Q 的两倍，画一条切线并找出另一交点 S，则 $Q+Q=2Q=-S$。

利用前述的运算规则，可以证明集合 $E(a,b)$ 是 Abel 群。

加法的代数描述

在本小节中，我们给出一些用于椭圆曲线上加法的结论①。对不是互为负元的两个不同的点 $P=(x_P,y_P)$ 和 $Q=(x_Q,y_Q)$，连接它们的曲线 l 的斜率 $\Delta=(y_Q-y_P)/(x_Q-x_P)$。$l$ 恰与椭圆曲线相交与另一点，即 P 与 Q 之和的负元。利用某些代数运算，我们可如下表示和 $R=P+Q$：

$$\begin{aligned} x_R &= \Delta^2 - x_P - x_Q \\ y_R &= -y_P + \Delta(x_P - x_R) \end{aligned} \tag{10.3}$$

我们也需要能够计算一个点与它自身相加：$P+P=2P=R$。当 $y_P\neq 0$ 时，该表达式为

$$\begin{aligned} x_R &= \left(\frac{3x_P^2+a}{2y_P}\right)^2 - 2x_P \\ y_R &= \left(\frac{3x_P^2+a}{2y_P}\right)(x_P - x_R) - y_P \end{aligned} \tag{10.4}$$

例如,取 $p=23$。考虑椭圆曲线方程 $y^2=x^3+x+1$,这里 $a=b=1$。注意,该方程与图 10.4(b)中的方程是相同的。图中显示了所有满足方程的实点。对 $E_{23}(1,1)$,我们只对如下非负整数感兴趣,它们位于从(0,0)到($p-1$,$p-1$)的象限中,满足模 p 的方程。表 10.1 列出了若干点(除 O 外),这些点是 $E_{23}(1,1)$的一部分,图 10.5 给出了 $E_{23}(1,1)$上的点。注意这些点除了一个之外,均关于 $y=11.5$ 对称。

表 10.1　椭圆曲线 $E_{23}(1,1)$上的点

(0,1)	(6,4)	(12,19)
(0,22)	(6,19)	(13,7)
(1,7)	(7,11)	(13,16)
(1,16)	(7,12)	(17,3)
(3,10)	(9,7)	(17,20)
(3,13)	(9,16)	(18,3)
(4,0)	(11,3)	(18,20)
(5,4)	(11,20)	(19,5)
(5,19)	(12,4)	(19,18)

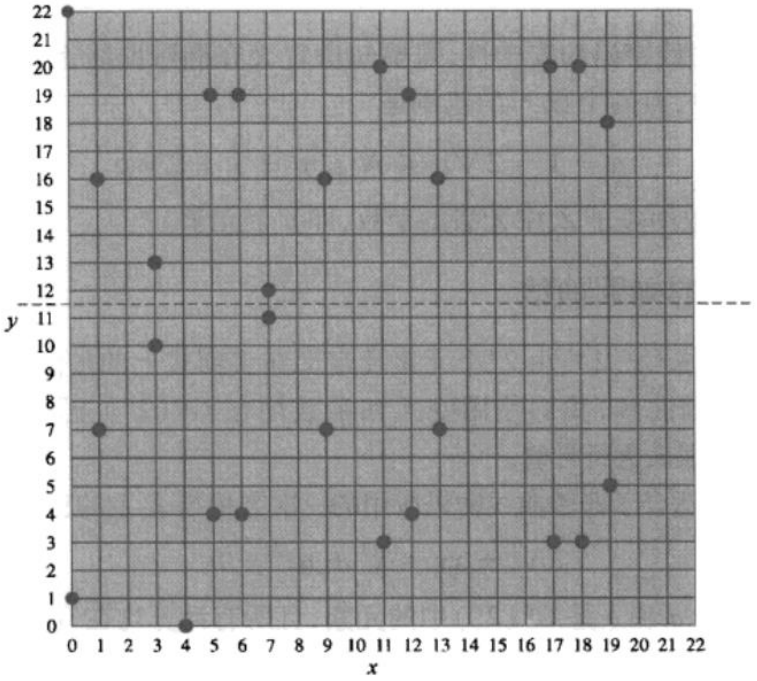


图 10.5　椭圆曲线 $E_{23}(1,1)$

$E_p(a,b)$上的加法运算构造与定义在实数上的椭圆曲线中描述的代数方法是一致的。对任何点 $P,Q\in E_p(a,b)$

(1) $P+O=P$。

(2) 若 $P=(x_P,y_P)$,则 $P+(x_P,-y_P)=O$。点$(x_P,-y_P)$是 P 的负元,记为 $-P$。例如,对 $E_{23}(1,1)$上的点 $P=(13,7)$,有 $-P=(13,-7)$,而 $-7 \bmod 23=16$,因此,$-P=(13,16)$,该点也在 $E_{23}(1,1)$上。

（3）若 $P=(x_P,y_P)$，$Q=(x_Q,y_Q)$，且 $P\neq -Q$ 则 $R=P+Q=(x_R,y_R)$ 由下列规则确定：

$$x_R=(\lambda^2-x_P-x_Q)\bmod p$$

$$y_R=(\lambda(x_P-x_R)-y_P)\bmod p$$

其中

$$\lambda=\begin{cases}\left(\dfrac{y_Q-y_P}{x_Q-x_P}\right)\bmod p, & P\neq Q\\ \left(\dfrac{3x_P^2+a}{2y_P}\right)\bmod p, & P=Q\end{cases}$$

（4）乘法定义为重复相加。如 $4P=P+P+P+P$。

例如取 $E_{23}(1,1)$ 上的 $P=(3,10)$，$Q=(9,7)$，那么

$$\lambda=\left(\frac{7-10}{9-3}\right)\bmod 23=\left(\frac{-3}{6}\right)\bmod 23=\left(\frac{-1}{2}\right)\bmod 23=11$$

$$x_R=(11^2-3-9)\bmod 23=109\bmod 23=17$$

$$y_R=(11(3-17)-10)\bmod 23=-164\bmod 23=20$$

所以 $P+Q=(17,20)$。为计算 $2P$，先求

$$\lambda=\left(\frac{3(3^2)+1}{2\times 10}\right)\bmod 23=\left(\frac{5}{20}\right)\bmod 23=\left(\frac{1}{4}\right)\bmod 23=6$$

上述等式的最后一步中需求 4 在 Z_{23} 中的乘法逆元。这可以用 4.4 节定义的扩展 Euclid 算法实现。注意到 $(6\times 4)\bmod 23=24\bmod 23=1$。

$$x_R=(6^2-3-3)\bmod 23=30\bmod 23=7$$

$$y_R=(6(3-7)-10)\bmod 23=(-34)\bmod 23=12$$

可见 $2P=(7,12)$。

为了确定各种椭圆曲线密码的安全性，需要知道定义在椭圆曲线上的有限 Abel 群中点的个数。在有限群 $E_p(a,b)$ 中，点的个数 N 的范围是

$$p+1-2\sqrt{p}\leqslant N\leqslant p+1+2\sqrt{p}$$

所以，$E_p(a,b)$ 上点的个数约等于 Z_p 中元素的个数，即 p 个元素。

则能够学习以下 4 个方向的内容：

对称加密

- **DES 加密算法**
- **AES 加密算法**

哈希函数

- **SHA2 哈希函数**
- **SHA3 哈希函数**

数字签名：

- 比特币的 **ECDSA** 签名算法
- 以太坊的 **EdDSA** 签名算法
- 门罗币的环签名算法

公钥加密：

- **ElGamal** 加密
- **ECIES** 加密
- 门罗币的同态加密算法
- **Paillier** 同态加密