密码学多签系列

第 5 课：Li17 两方签名与密钥刷新

lynndell 博士

新火科技 密码学专家 lynndell2010@gmail.com

目录

密码学基础系列

1. 对称加密与哈希函数
2. 公钥加密与数字签名
3. RSA、环签名、同态加密
4. 承诺、零知识证明、BulletProof 范围证明、Diffie-Hellman 密钥协商

多签系列

5. Li17 两方签名与密钥刷新
6. GG18 多方签名
7. GG20 多方签名
8. CMP20 多方签名
9. DKLs18 两方/20 多方签名
10. Schnorr/EdDSA 多方签名

zk 系列

11. Groth16 证明系统
12. Plonk 证明系统
13. UltraPlonk 证明系统
14. SHA256 查找表技术
15. Halo2 证明系统
16. zkSTARK 证明系统

1.预备知识

1.1 Paillier 同态加密

困难假设 1：因子分解困难问题

对于两个长度相等的大素数 p,q， $p \neq q$，计算 $N = p \cdot q$。公开 N，求 p,q 是困难的。

需要指数时间暴力搜索，在多项式时间内不可行。

困难假设 2：判决复合冗余 Decisional Composite Residuosity (DCR)困难问题

不存在概率多项式时间攻击者 $\mathcal{A}$ 能够以不可忽略的优势概率区分以下 2 个分布

$$\left\{[N,c], c = r^N \bmod N^2\right\}, \left\{[N,c], c = g^m \cdot r^N \bmod N^2\right\}$$

其中， $m, r \in \mathbb{Z}_N, N = p \cdot q, g = N+1$。

密钥生成：生成两个长度相同的**大素数** p,q，满足 $\gcd\left(pq,(p-1)(q-1)\right) = 1$，该性质确保这两个素数长度相同；计算 $N := pq$，最小公倍数 $\lambda = lcm(p-1,q-1)$；分式除法函数 $L(y) = (y-1)/N$；选择正整数 $g = 1+N \in Z_{N^2}^*$，使得 $\mu = \left(L(g^\lambda \bmod N^2)\right)^{-1} \bmod N$ 存在。

公钥为 N，私钥为 p,q 或 λ。

加密：消息 $m \in Z_N$，选择随机数 $r \in Z_N^*$，计算密文 $c := g^m \cdot r^N \bmod N^2$。

解密：输入密文 $c \in Z_{N^2}$，如下计算解密 $m := L(c^\lambda \bmod N^2) \cdot \mu \bmod N$。

$$c = g^m \cdot r^n \bmod n^2$$
$$c^\lambda \bmod n^2 = g^{\lambda m} \cdot r^{\lambda n} \bmod n^2 = g^{\lambda m} \cdot 1 \bmod n^2 = (1+n)^{m\lambda} \bmod n^2 = 1 + nm\lambda \bmod n^2$$
$$g^\lambda \bmod n^2 = (1+n)^\lambda \bmod n^2 = 1 + n\lambda \bmod n^2$$
$$L(c^\lambda \bmod n^2) = \frac{c^\lambda \bmod n^2 - 1}{n} = m\lambda \bmod n^2$$
$$L(g^\lambda \bmod n^2) = \frac{g^\lambda \bmod n^2 - 1}{n} = \lambda \bmod n^2$$
$$m = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n$$

同态性：给定两个密文 $c_1, c_2 \in Z_{N^2}$， $c_1 = Enc_{pk}(m_1), c_2 = Enc_{pk}(m_2)$

- 定义**密文同态加法** $\oplus$

$$c_1 \oplus c_2 = c_1 c_2 \bmod N^2 = g^{m_1+m_2} \cdot (r_1 r_2)^N \bmod N^2$$

因此，$c_1 \oplus c_2 = c_1 c_2 \bmod N^2 = Enc_{pk}(m_1 + m_2 \bmod N)$。

- 给定 $a \in Z_N, c = Enc_{pk}(m)$，定义**随机数与密文的同态乘法**$\otimes$：

$$a \otimes c = c^a \bmod N^2 = g^{am} \cdot (r^a)^N \bmod N^2 = Enc_{pk}(a \cdot m \bmod N)$$

1.2 ECDSA

初始化：椭圆曲线生成元为G，群的阶$|F_r|$；标量域为F_r，基域为F_q。

密钥生成：输入安全参数λ，输出私钥$x \in F_r$和公钥PK，且满足以下离散对数关系

$$PK = x \cdot G$$

签名：输入任意消息M，计算$m := Hash(M)$；选择随机数$k \in F_r$，计算$R := k \cdot G$，取R横坐标为$r := R_x \bmod |F_r|$；计算$s := k^{-1}(m + xr) = (k^{-1}m + k^{-1}xr) \bmod |F_r|$，则签名为$(r, s)$。

ECDSA 目标 1 计算： $R := k \cdot G$

ECDSA 目标 2 计算： $s := k^{-1}(m + xr) = k^{-1}m + k^{-1}xr$

验证：输入消息M，计算$m := Hash(M)$；校验$r, s \in F_r$，计算$R' := (s^{-1}m) \cdot G + (s^{-1}r) \cdot PK$，取$R'$横坐标为$r' := R'_x \bmod |F_r|$；校验$r == r'$。如果相等，则接受，否则拒绝。

公式推导过程如下：

$$\begin{aligned}
R' &= (s^{-1}m) \cdot G + (s^{-1}r) \cdot PK \\
&= (s^{-1}m) \cdot G + (s^{-1}rx) \cdot G \\
&= \left(s^{-1}(m + rx)\right) \cdot G \\
&= k \cdot G
\end{aligned}$$

ECDSA 的验证本质：

$$\begin{aligned}
&s = k^{-1}(m + xr) \\
&k = s^{-1}(m + xr) \\
&k \cdot G = s^{-1}m \cdot G + s^{-1}xr \cdot G \\
&R = s^{-1}m \cdot G + s^{-1}r \cdot PK
\end{aligned}$$

检测$(r, |F_r| - s)$是否为合法的签名：

$$|F_r|-s=k^{-1}(m+xr)$$
$$k(|F_r|-s)=(m+xr)$$
$$k(|F_r|-s)\cdot G=m\cdot G+xr\cdot G$$
$$-ks\cdot G=m\cdot G+r\cdot PK$$
$$-R=s^{-1}m\cdot G+s^{-1}r\cdot PK$$

计算出$-R$，**纵坐标是负的无所谓，**取横坐标得到的就是$r':=R'_x \bmod |F_r|$， 校验$r==r'$。如果相等，则接受，否则拒绝。因此，$(r,|F_r|-s)$是合法签名。既然有 2 个合法签名，所以 Li17/BTC/ETH 等系统均计算$s=\min\{s',|F_r|-s'\}$，两个签名，确定一个小的作为正确签名，另外一个大的是错误签名。

1.3 零知识证明

1.3.1 zk-Schnorr 证明知道 ECC 私钥

zk-Schnorr 证明协议 A 版

初始化：椭圆曲线生成元为G，标量域为F_r，基域为F_q；

证明方的私钥为sk，公钥为PK，满足离散对数关系$PK=sk\cdot G$。

1：（**承诺**）选择随机数$r\in F_r$，计算$R:=r\cdot G$；

2：（**挑战**）计算随机数$c:=hash(PK,R) \bmod |F_r|$；

3：（**响应**）计算$z:=r+c\cdot sk \bmod |F_r|$，发送$(R,z)$；

4：（**验证**）计算随机数$c:=hash(PK,R) \bmod |F_r|$，校验$z\cdot G==R+c\cdot PK$。

公式推导：

$$z\cdot G=(r+c\cdot sk)\cdot G=R+c\cdot PK$$

zk-Schnorr 证明协议 B 版

证明方的私钥为sk，公钥为PK，满足离散对数关系$PK=sk\cdot G$。

1：（**承诺**）选择随机数$r\in F_r$，计算$R:=r\cdot G$；

2：（**挑战**）计算随机数$c:=hash(PK,R) \bmod |F_r|$；

3：（**响应**）计算$z:=r+c\cdot sk \bmod |F_r|$，发送$(c,z)$；

4：（**验证**）计算$R:=z\cdot G-c\cdot PK$，校验$c==hash(PK,R) \bmod |F_r|$。

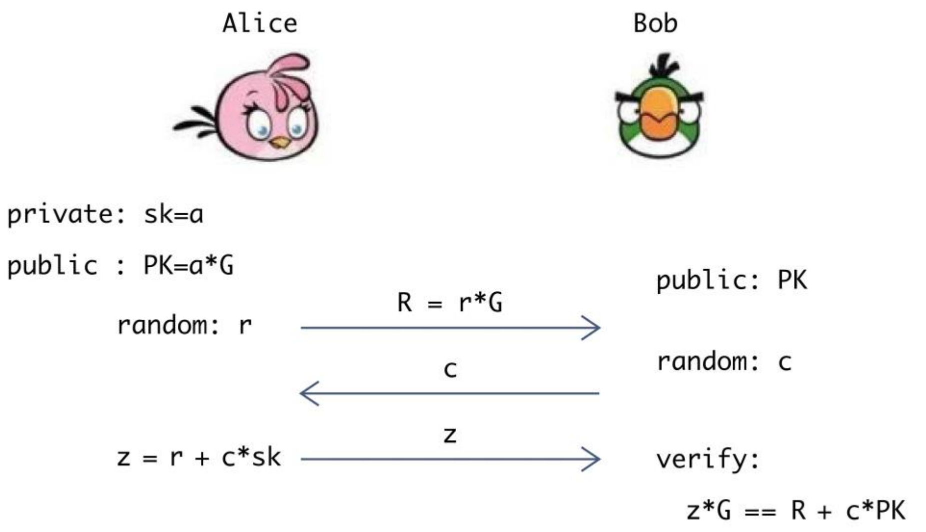

图 1. 交互式 Schnorr 协议

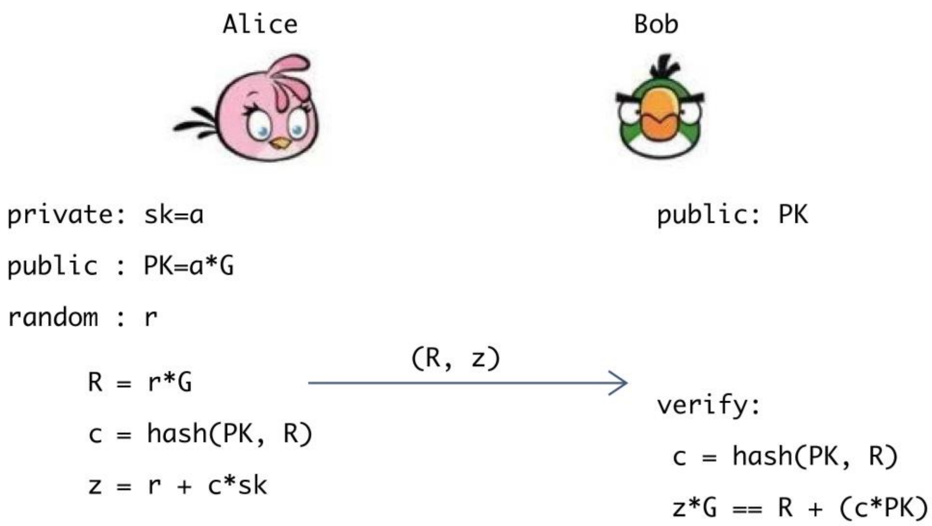

图 2. 非交互式 Schnorr 协议 **A** 版

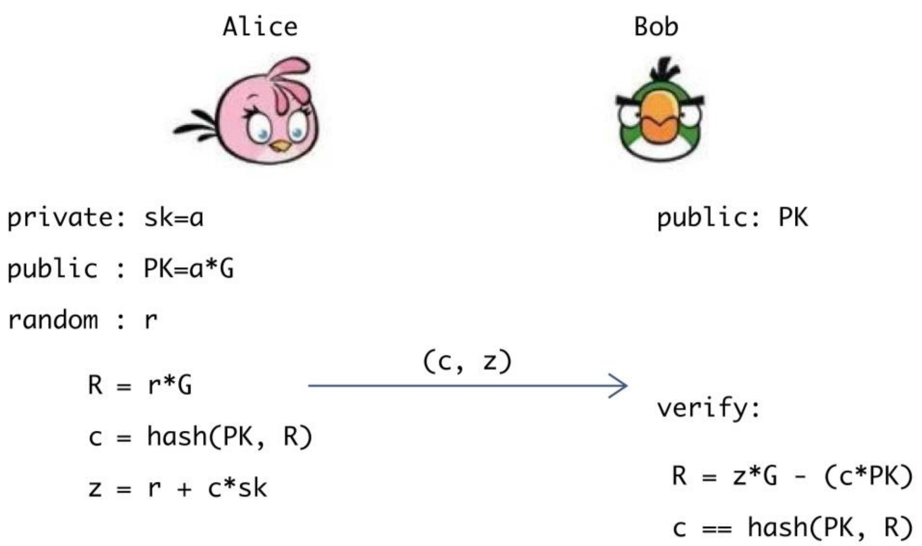

图 3. 非交互式 Schnorr 协议 **B** 版

1.3.2 zk-Paillier-N 证明知道 Paillier 私钥

1.3.2.1 预备知识 1：初等数论

（1）原根 g 存在性

设 g 为模数 p 的一个原根，原根满足 $g^{\varphi(p)} \equiv 1 \bmod p$，则 $g^x, g^{x+1}, \ldots, g^{x+p-2}$ 在模 p 对应 [1,p-1]中的每一项。

素数 p 必定有原根 g。

原根 g 的求解方法：

根据费马小定理 $g^{p-1} \equiv 1 \bmod p$，则枚举 g；

然后枚举 p-1 的质因子 Δ；如果 $g^{\frac{p-1}{\Delta}} \equiv 1 \bmod p$，则不是原根，否则原根。

分析：①如果 $g^{\frac{p-1}{\Delta}} \equiv 1 \bmod p$，则 $\frac{p-1}{\Delta} < \varphi(p)$，与原根定义 $\varphi(p)$ 最小矛盾。

②原根通常很小，枚举 g 是可行的。

（2）BSGS 算法（Baby-Step Giant-Step）

g 与 p 互素，g 是原根，则同余方程 $g^t \equiv a \bmod p$，能够快速求 t，计算复杂度 $O(\sqrt{p})$。

求解方法：选择 $A, B \in \left[0, \sqrt{p}\right]$，令 $t = A\left\lceil \sqrt{p} \right\rceil - B$ 带入同余方程

$$g^{A\lceil \sqrt{p} \rceil - B} \equiv a \bmod p$$

同余方程转换为

$$g^{A\lceil \sqrt{p} \rceil} \equiv a g^B \bmod p$$

已知 g, a，枚举 A, B 计算同余方程两边的取值，存入表中。如果表中的值发生碰撞，则找到方程的解。因此，计算复杂度为 $O(\sqrt{p})$。

（3）同余方程

定理：如果 $\gcd(N,(p-1)) = 1$，则同余方程 $Ny_1 \equiv t \bmod (p-1)$ 有唯一解。

证明：因为 $\gcd(N,(p-1)) = 1$，所以根据**欧拉定理**

$$N^{\varphi(p-1)} \equiv 1 \bmod (p-1)$$
$$N^{\varphi(p-1)-1+1} \equiv 1 \bmod (p-1)$$
$$N \cdot N^{\varphi(p-1)-1} \equiv 1 \bmod (p-1)$$
$$N \cdot N^{-1} \equiv 1 \bmod (p-1)$$

则存在**模反逆元** $N^{-1}=N^{\varphi(p-1)-1}$。**模反逆元** N^{-1} 满足 $NN^{-1}\equiv 1\bmod (p-1)$，令 $y_1=N^{-1}t\bmod (p-1)$，则 y_1 是方程 $Ny_1\equiv t\bmod (p-1)$ 的一个解。如果还有另外一个解 y_1'，则 $Ny_1\bmod (p-1)\equiv t\bmod (p-1)\equiv Ny_1'\bmod (p-1)$。因为 $\gcd(N,(p-1))=1$，所以 $y_1\equiv y_1'\bmod (p-1)$ 两个解相等。因此，解是唯一的。

（4）N 次剩余

令 $m\ge 2,\gcd(a,p)=1,N\ge 2$，如果同余方程 $x^N\equiv a\bmod p$ 有解 x，则称 a 是模 p 的 N 次剩余，否则 a 是模 p 的 N 次非剩余。

定理：设原根为 g，则同余方程 $x^N\equiv a\bmod p$ 能够求解，且解唯一。

求解方法： 设 $x_1\bmod p$ 是同余方程 $x^N\equiv a\bmod p$ 的解，由于 $\gcd(a,p)=1$，则 $\gcd(x_1^N,p)=1$，则 $\gcd(x_1,p)=1$。因此，存在 y_1 满足

$$x_1\equiv g^{y_1}\bmod p$$

则有

$$x_1^N=g^{Ny_1\bmod (p-1)}\equiv a\bmod p$$

构造同余方程 $g^t\equiv a\bmod p$，使用 **Baby-Step Giant-Step 算法**计算唯一解 t。

所以有

$$g^{Ny_1\bmod (p-1)}\equiv g^t\bmod p$$

因此，同余方程 $Ny_1\equiv t\bmod (p-1)$ 求解唯一 y_1 后，则能够求解唯一 x_1。

1.3.2.2 预备知识 2：n^s 次方证明协议

双方公共输入为 n,s,u；

证明方知道秘密 v，满足关系 $u=v^{n^s}\bmod n^{s+1}$，或证明知道 u 的 n^s 次根是 v。

证明方	验证方
承诺： 选择随机数 $r\in\{0,\dots,n^{s+1}\}$，计算 $a=r^{n^s}\bmod n^{s+1}$；发送 a	
	挑战： 选择 k-bit 的随机数 e；发送 e
响应： 计算 $z:=rv^e\bmod n^{s+1}$；发送 z	

	校验： $z^{n^s} == au^e \bmod n^{s+1}$

分析：看作 Sigma 协议的具体实例，都是 4 步骤：**承诺、挑战、响应、验证。**

公式推导：$au^e \bmod n^{s+1} = r^{n^s} v^{n^s \cdot e} \bmod n^{s+1} = (rv^e)^{n^s} \bmod n^{s+1} = z^{n^s}$

1.3.2.3zk-Paillier-N 证明知道 Paillier 私钥

或证明拥有正确的 Paillier 密钥对，即 N 与 $\varphi(N)$ 互素，$\gcd(N,\varphi(N))=1$。

证明方	验证方
生成 Paillier 私钥 $\lambda=\varphi(N)$ 和公钥 N。 **发送** Paillier 公钥 N。	
	选择随机数 y，基于 Paillier 公钥 N 计算 $x=y^N \bmod N^2$；使用 n^s **次方证明协议 (s=1)**证明其知道 x 的 N 次根是 y，生成 $proof$；发送 $x, proof$；
校验 $proof$；因为 $\gcd(N,\varphi(N))=1$，所以可以 **N 次剩余求解，**使用 Paillier 私钥 $\varphi(N)$ 计算 x 的 N 次根 y'； 分析：如果不知道 Paillier 私钥，无法 N 次剩余求解；如果 Paillier 密钥对错误，则 N 次剩余无法求解。 发送 y'。	
	接收 y'；校验 $y'==y$。校验成功，则确保对方知道 Paillier 私钥。

分析：看作 Sigma 协议的具体实例，都是 4 步骤：**承诺、挑战、响应、验证。**

1.2 1.3.3 zk-Paillier-Enc 证明加密 ECC 私钥且 ECC 私钥范围正确

ECC 私钥 256bit；Paillier 加密 2048bit 数据，空间更大。

初始化：椭圆曲线生成元为 G，标量域为 F_r，基域为 F_q；

证明方拥有的保密信息为 Paillier 私钥 sk 和 ECC 私钥 x_1，满足运算关系

$$c_{key} = Enc_{pk}(x_1), Q_1 = x_1 \cdot G, x_1 \in F_r$$

证明方	验证方
①生成 Paillier 密钥对为(pk,sk)和 ECC 密钥对(x_1,Q_1)，其中$Q_1=x_1\cdot G$； ②计算 Paillier 加密$c_{key}=Enc_{pk}(x_1)$； **发送**c_{key},pk,Q_1；	
	接收c_{key},pk,Q_1； ①选择2个随机数$a\in F_r,b\in F_{r^2}$，计算 Paillier 加密$c_b:=Enc_{pk}(b)$； ②同态计算 $c':=(a\otimes c_{key})\oplus c_b=Enc_{pk}(ax_1+b)$ ③计算承诺与打开承诺$(C_1,D_1):=Com(a,b)$ ④计算$Q':=a\cdot Q_1+b\cdot G$； 分析：（1）c_{key}关联x_1； （2）$Q'=a\cdot Q_1+b\cdot G=(ax_1+b)\cdot G$证明方的公钥$Q_1$关联$x_1$，且$Q'$对应的私钥为$ax_1+b$。 **发送**密文$c'$和承诺$C_1$；
接收密文c'和承诺C_1； ② 解密$\alpha:=Dec_{sk}(c')=ax_1+b$ ②计算$\hat{Q}:=\alpha\cdot G=(ax_1+b)\cdot G$； 分析：$\hat{Q}$关联的$x_1$来自$c_{key}$； ③计算承诺与打开承诺$(C_2,D_2):=Com(\hat{Q})$ 发送承诺C_2；	

	接收承诺C_2 **发送**打开承诺D_1；
获得打开承诺a,b； ①校验$\alpha == a \cdot x_1 + b$； ②**zk 范围证明：** $ZK\{x_1 \| x_1 \in F_r\}$，生成$proof$；（Paillier 能够加密 1024bit，而离散对数中的x_1仅 256bit）。 **发送**打开承诺D_2和$proof$	
	接收$\hat{Q}$和$proof$； 校验$proof$，确保x_1范围正确； $\hat{Q} == Q'$，确保c_{key}中的x_1等于Q_1中的x_1； 分析：左边$\hat{Q}$关联的私钥x_1来自c_{key}， 右边$Q' = a \cdot Q_1 + b \cdot G$关联私钥$x_1$来自$Q_1$；

分析：看作 Sigma 协议的扩展版

1.3 1.3.4 zk-RangeProof 范围证明【基础版】

Boudot F. Efficient proofs that a committed number lies in an interval[C]//Eurocrypt. 2000, 1807: 431-444.
Section1.2.2

说明：基础版有负数空间，升级版是正数空间。
应用场景：大数的范围证明，发表于 2000，而不是 ECC 的 Pedersen 承诺信息中的 BulletProof 2018。BlletProof 2018 的 size 更短。使用向量内积承诺计算折半响应，降低 size。

初始化：1024bit 的大素数p，1023bit 的大素数q，且$q \mid p-1$。g,h为群的生成元，阶为q。秘密x的承诺为$E = E(x,r) = g^x \cdot h^r \bmod p$，其中，随机数$r \in Z_p^*$。

Alice 秘密为x，证明秘密属于某个范围$x \in [-b, 2b]$（有负数空间）。

证明方	验证方
承诺：①选择正随机数$\omega_1 \in [0,b]$，其中b为 512bit，计算负随机数 $$\omega_2 := \omega_1 - b \in [-b,0]$$ ②选择两个随机数$\eta_1,\eta_2 \in [0,q-1]$，计算 2 个 Pedersen 承诺 $$W_1 := g^{\omega_1} h^{\eta_1} \bmod p$$ $$W_2 := g^{\omega_2} h^{\eta_2} \bmod p$$ 发送 Pedersen 承诺W_1,W_2。	
	挑战：发送随机位$c \in \{0,1\}$；
响应： ①如果$c == 0$，则发送打开 Pedersen 承诺$\omega_1,\omega_2,\eta_1,\eta_2$； ②如果 $c == 1$，则对 $x \in [-b,2b]$ 寻找 $j \in \{1,2\}$，满足范围$x + \omega_j \in [0,b]$，计算响应$u := x + \omega_j, v := r + \eta_j$，发送$u,v$； **分析：**存下以下 3 种情况：对于 有负数空间的 $x \in [-b,2b]$ 如果 $x \in [-b,0]$，则需要使用 $\omega_1 \in [0,b]$，使得 $x + \omega_j \in [0,b]$； 如果 $x \in [b,2b]$，则需要使用 $\omega_2 \in [-b,0]$，使得 $x + \omega_j \in [0,b]$； 如果 $x \in [0,b]$，则需要使用 $\omega_1 \in [0,b]$ 或 $\omega_2 \in [-b,0]$，使得 $x + \omega_j \in [0,b]$。	
	验证：

	①如果 $c == 0$，则接收 Pedersen 打开承诺 $\omega_1, \omega_2, \eta_1, \eta_2$，校验 Pedersen 打开承诺 $W_1 == g^{\omega_1} h^{\eta_1} \bmod p, W_2 == g^{\omega_2} h^{\eta_2} \bmod p$ 作用：Pedersen 打开承诺一致性，确保 ω_1, ω_2 范围是正确的，且没泄露秘密 x, r； ②如果 $c == 1$，则接收到 u, v，则校验承诺 $E \cdot W_j == g^u h^v \bmod p$,**范围校验** $u \in [0, b]$。 **公式推导：** $g^u \cdot h^v = g^{x+\omega_j} \cdot h^{r+\eta_j}$ $= (g^x h^r) \cdot (g^{\omega_j} h^{\eta_j}) = E \cdot W_j$ 作用：范围校验 $u \in [0, b]$ 确保 x 的范围正确，且随机数 ω_j 对 x 随机化，实现零知识。

分析：看作 Sigma 协议的具体实例，都是 4 步骤：**承诺、挑战、响应、验证。**
分析： $c == 0$ 确保随机数 ω_j 范围正确性，$c == 1$ 确保 x 范围正确性。该协议**并行运行 t=40 次**，则证明方作弊且全都成功的概率为 2×2^{-t}，概率忽略。
反之，校验 **t=40** 次全正确，则证明方每次都诚实执行协议，且 x 范围正确。

1.4 1.3.5 zk-RangeProof*范围证明【升级版】

令 $l = \lfloor q/3 \rfloor$，$x \in \{0, ..., l\}$。双方知道 $|F_r|, l = |F_{r/3}|, t = 40$ 次，令 $i = \{1, ..., t\}$；

zk 范围证明 $zk\{x \in F_r\}$（正数空间）。

证明方	验证方
①生成 Paillier 密钥对 $(N, \varphi(N))$； ②选择随机数 $r_0 \in \mathbb{Z}_n$，对 ECC 私钥 x 进行 Paillier 加密 $c = Enc_{pk}(x, r_0)$；发送 (c, N)	
	选择位宽为 t 的随机数 $e \leftarrow \{0,1\}^t, e = \{e_1, ..., e_t\}$，计算承诺与打开承诺 $(C_1, D_1) := Com(e)$，发送承诺 C_1；

①选择 t 个大随机数 $w_1^1,...w_1^t \leftarrow \{l,...,2l\}$， 计算 t 个小随机数 $w_2^i := w_1^i - l \in \{0,...,l\}$； ②将大小随机数 w_1^i, w_2^i 以 1/2 的概率进行随机互换 $w_1^i \rightleftarrows w_2^i$； ③选择随机数 $r_1^i, r_2^i \in Z_N$，计算 Paillier 加密 $c_1^i := Enc_{pk}(w_1^i, r_1^i),$ $c_2^i := Enc_{pk}(w_2^i, r_2^i)$ **发送**密文承诺 c_1^i, c_2^i。	
	接收 c_1^i, c_2^i，则发送打开承诺 D_1；
接收 $e = \{e_1,...,e_t\}$。 ①如果 $e_i == 0$，则**发送** $z_i := (w_1^i, r_1^i, w_2^i, r_2^i)$； ②如果 $e_i == 1$，则寻找 $j \in \{1,2\}$ 使得 $u = x + w_j^i \in \{l,...,2l\}$ 成立，**发送** $z_i := (j, u, r_0 \cdot r_j^i \bmod N)$。 分析：$w_1^i \leftarrow \{l,...,2l\}, w_2^i \leftarrow \{0,...,l\}$ 且 $x \in \{0,...,l\}$，所以存在 $u \in \{l,...,2l\}$	
	根据 e_i 解析出 z_i。 ①如果 $e_i == 0$，则接收到 $z_i = (w_1^i, r_1^i, w_2^i, r_2^i)$，**校验密文承诺** $c_1^i == Enc_{pk}(w_1^i, r_1^i)$ $c_2^i == Enc_{pk}(w_2^i, r_2^i)$ 且 (w_1^i, w_2^i) 中的一个属于 $\{l,...,2l\}$，另一个属于 $\{0,...,l\}$；作用：承诺打开一致性，

	两个随机数都打开，确保 (w_1^i, w_2^i) 范围是正确的。 ② 如果 $e_i == 1$，则接收到 $z_i = (j, u, r_0 \cdot r_j^i \bmod N)$，则**同态校验** $c \oplus c_j^i = Enc_{pk}(x, r_0) \oplus Enc_{pk}(w_j^i, r_j^i)$ $= Enc_{pk}(u, r_0 \cdot r_j^i)$ 且**范围校验** $u \in \{l, ..., 2l\}$。 作用：（1）paillier 同态校验，确保打开承诺正确；（2）u 范围正确，则确保 x 的范围正确性。（3）随机数 ω_j 是对 x 随机化，实现零知识。

分析：看作 Sigma 协议的扩展版

该协议**并行运行 t=40 次**，则证明方作弊且全都成功的概率为 2×2^{-t}，概率可忽略。反之，如果校验 t=40 次全正确，则证明方每次都诚实执行协议，且 x 范围正确。

1.4 Diffie-Hellman 密钥交换系列

1.5 1.4.1 Diffie-Hellman 密钥交换（诚实版）

Alice	Bob
生成私钥 $x_1 \in F_r$，计算公钥 $Q_1 = x_1 \cdot G$； **发送** Q_1；	生成私钥 $x_2 \in F_r$，计算公钥 $Q_2 = x_2 \cdot G$； **发送** Q_2；
接收 Q_2，计算 $Q_{common} := x_1 \cdot Q_2$	接收 Q_1，计算 $Q_{common} := x_2 \cdot Q_1$
协商结果：$Q_{common} := x_1 x_2 \cdot G$	

因此，Alice 与 Bob 计算出相同的**公共密钥** $Q_{common} = x_1 x_2 \cdot G$，也能计算用于对称加密的**会话密钥** $key = hash(nonce, Q_{common})$。

如果 Q 公开，则称为**公共公钥**；对应的 $x_1 x_2$ 称为**公共私钥**（不出现）。

Alice	Bob
生成私钥 $x_1 \in F_r$，计算公钥 $Q_1 = x_1 \cdot G$；	生成私钥 $x_2 \in F_r$，计算公钥 $Q_2 = x_2 \cdot G$；

发送 Q_1；	**发送** Q_2；
接收 Q_2，计算 $Q_{common}:=Q_1+Q_2$	接收 Q_1，计算 $Q_{common}:=Q_1+Q_2$
公共公钥 $Q_{common}:=Q_1+Q_2=(x_1+x_2)\cdot G$ 公共私钥 $x_{common}=x_1+x_2$ **不出现**	

上述协议要求双方均诚实！如果有一方不诚实。如 Alice 不诚实，对后续不利。

1.6 1.4.2 Diffie-Hellman 密钥交换（非诚实版）

Alice	**Bob**
被黑客攻击，选择随机点 Q_1 作为公钥，不知道 (Q_1,G) 离散对数关系 x_1；**发送** Q_1；	生成私钥 $x_2\in F_r$，计算公钥 $Q_2=x_2\cdot G$； **发送** Q_2；
接收 Q_2，无法计算 $Q_{common}:=x_1\cdot Q_2$	接收 Q_1，计算 $Q_{common}:=x_2\cdot Q_1$

因此，Alice 无法计算**公共密钥** $Q_{common}=x_1x_2\cdot G$，也无法计算用于对称加密的**会话密钥** $key=hash\left(nonce,Q_{common}\right)$。因此，Alice 与 Bob 后续不能保密通信，也不能基于**公共密钥** Q_{common} 计算其他信息。

1.7 1.4.3Diffie-Hellman 密钥交换（强制诚实版）

Alice	Bob
私钥 $x_1\in F_r$，公钥 $Q_1=x_1\cdot G$；	私钥 $x_2\in F_r$，公钥 $Q_2=x_2\cdot G$；
zk-Schnorr 证明知道私钥 x_1，生成 $proof_1$。 发送 $(proof_1,Q_1)$	**zk-Schnorr 证明**知道私钥 x_2，生成 $proof_2$。 发送 $(proof_2,Q_2)$
校验：$(proof_2,Q_2)$ 有效性，然后计算 $Q_{common}:=x_1\cdot Q_2=x_1x_2\cdot G$；	**校验：**$(proof_1,Q_1)$ 有效性，然后计算 $Q_{common}:=x_2\cdot Q_1=x_1x_2\cdot G$；

因此，Alice 和 Bob 均一定能计算**公共密钥** $Q_{common}=x_1x_2\cdot G$，也一定能计算用于对称加密的**会话密钥** $key=hash\left(nonce,Q_{common}\right)$。因此，Alice 与 Bob 后续一定能能保密通信，也一定能能基于**公共密钥** Q_{common} 计算其他信息。

1.5 理想函数 F

协议一：存在可信第三方 $TrustParty$

Alice	可信第三方 T	Bob
用可信第三方公钥**加密**私钥 x_1 并**发送**给可信第三方 T;		用可信第三方公钥**加密**私钥 x_2 并**发送**给可信第三方 T;
	接收，解密获得 x_1 和 x_2； 计算**公共密钥** $x_1x_2\cdot G$，使用 Alice 和 Bob 的公钥加密并发送给双方	
解密获得**公共密钥**为 $Q_{common}=x_1x_2\cdot G$。		解密获得**公共密钥**为 $Q_{common}=x_1x_2\cdot G$。

协议二：不存在可信第三方 $NoTrustParty$（**Diffie-Hellman 密钥交换协议强制诚实版**）

Alice	Bob
私钥 $x_1\in F_r$，公钥 $Q_1=x_1\cdot G$；	私钥 $x_2\in F_r$，公钥 $Q_2=x_2\cdot G$；
zk-Schnorr 证明知道私钥 x_1，生成 $proof_1$。 发送 $(proof_1,Q_1)$	zk-Schnorr 证明知道私钥 x_2，生成 $proof_2$。 发送 $(proof_2,Q_2)$
校验 $proof_2$ 有效性，然后计算 $Q_{common}:=x_1\cdot Q_2=x_1x_2\cdot G$	校验 $proof_1$ 有效性，然后计算 $Q_{common}:=x_2\cdot Q_1=x_1x_2\cdot G$

分析：协议 1 与协议 2 实现功能相同，但是协议 2 的安全性仅基于密码算法困难假设（更安全），而**协议 1 要额外依赖可信第三方，安全性不完备。**

协议三：**Diffie-Hellman** 理想函数 $\mathcal{F}_{Diffie-Hellman}$

理想函数 $\mathcal{F}$ 理解为以太坊上的智能合约（严格执行设定的规则，对该保密的数据保密，该发送的数据发送）。

Alice	**理想函数** $\mathcal{F}_{Diffie-Hellman}$	**Bob**
私钥 x_1 使用理想函数 F 的公钥加密并发送给 F		私钥 x_2 使用理想函数 F 的公钥加密并发送给 F
	接收，解密并计算**公共密钥** $x_1x_2 \cdot G$，使用 Alice 和 Bob 的公钥加密并发送给双方；	
获得 Diffie-Hellman **公共密钥**为 $Q_{common} = x_1x_2 \cdot G$		获得 Diffie-Hellman **公共密钥**为 $Q_{common} = x_1x_2 \cdot G$

协议二与协议三实现相同功能，安全性也相等。

分析：Diffie-Hellman 理想函数 $\mathcal{F}_{Diffie-Hellman}$ 是对可信第三方 $TrustParty$ 的**安全升级**。

目前有大量的**密码协议**（如承诺协议 Com、零知识证明协议 ZK、数字签名协议 Sig、公钥加密协议 $PKEnc$、不经意传输协议 $ObliviousTransfer$ 等）与理想函数 $\mathcal{F}_{com}, \mathcal{F}_{ZK}, \mathcal{F}_{sig}, \mathcal{F}_{PKEnc}, \mathcal{F}_{OT}$ 实现**相同功能，相等安全性。**

关键结论：
因此，调用理想函数等价于（黑盒子）调用对应的安全的密码协议。

使用理想函数 $\mathcal{F}$ 的三个原因：

1. **描述简洁性：**由于密码协议描述的步骤较多，为描述简洁，使用理想函数 $\mathcal{F}_{com}, \mathcal{F}_{ZK}, \mathcal{F}_{sig}, \mathcal{F}_{PKE}, \mathcal{F}_{OT}$ 替换描述简洁。
2. **用于协议设计：**设计安全的理想函数，有利于设计安全的密码协议/框架；**通用组合安全 Universally Composable Security：**子协议是安全的，设计一个大框架，多次调用子协议。如果大框架是安全的，则整个系统都安全。（目前有些协议单个运行安全，并行运行、并发运行不一定安全。这样的组合安全，达到的安全性非常高。）
3. **密码协议的安全性证明：**使用分布不可区分的概率模型，从攻击者角度，获得的全是随机数，攻击者无法提取有用信息。因此，能够用于证明协议/框架的安全性。

强制诚实版的 Diffie-Hellman 密钥交换协议：把零知识证明和公钥发送给对方。
强制诚实版的 Diffie-Hellman 密钥交换协议升级版(发起方数据加密发送，时刻防着对方)：

1. **（强制版）害怕对方不知道公钥对应的私钥，所以步骤 1 和 2 都要求双方 zk-Schnorr 证明知道私钥；**

2. （升级版）握手协议：检测对方是否能提供服务：**Alice** 发起方害怕自己数据丢失，所以对证明 $proof_1$ 和公钥 Q_1 承诺（等价于加密）发给理想函数 $\mathcal{F}_{com-zk}^{R_{DL}}$，接收到对方 **Bob** 的证明 $proof_2$ 和公钥 Q_2 后，理想函数 $\mathcal{F}_{com-zk}^{R_{DL}}$ 把数据发给对方。**[能实现安全性证明]**

2.两方密钥生成

	用户 P_1	服务方 P_2
1	①选择随机数 $x_1 \in F_{r/3}$，计算 $$Q_1 = x_1 \cdot G$$ 称为：分片私钥为 x_1，分片公钥为 Q_1 ②zk-Schnorr 证明知道分片私钥 x_1 $$proof_1 = ZK\{x_1 \mid Q_1 = x_1 \cdot G\}$$ ③对 Q_1 和 $proof_1$ 生成承诺与打开承诺 $[KGC_1, KGD_1] = Com(Q_1, proof_1)$ 发送承诺 KGC_1	
2		接收承诺 KGC_1 ①选择随机数 $x_2 \in F_r$，计算 $$Q_2 = x_2 \cdot G$$ 称为：分片私钥为 x_2，分片公钥为 Q_2 ②zk-Schnorr 证明知道分片私钥 x_2 $$proof_2 = ZK\{x_2 \mid Q_2 = x_2 \cdot G\}$$ 发送证明 $(proof_2, Q_2)$
3	接收证明 $proof_2$，校验，获得 Q_2；	

	①生成 Paillier 密码公钥和私钥(pk,sk)， 公钥长度为$\max\{3\log\|r\|+1,n\}$； ②计算 Paillier 加密$c_{key}:=Enc_{pk}(x_1)$为签名的同态计算做准备。 ③zk-Paillier-N 证明知道 Paillier 私钥 $proof_{Paillier,1}=ZK\left\{sk\middle\|pk=N=p_1\cdot p_2\right\}$ ④zk-Paillier-Enc 证明正确加密 ECC 私钥 $proof_{Paillier,2}=$ $ZK\left\{\begin{matrix}(c_{key},pk,Q_1,G)\\(x_1,r)\end{matrix}\middle\|\begin{matrix}c_{key}=Enc_{pk}(x_1,r),\\Q_1=x_1\cdot G\end{matrix}\right\}$ **发送** $KGD_1,pk,c_{key},proof_{Paillier,1},proof_{Paillier,2}$	
4		接收打开承诺KGD_1，获得Q_1和 $proof_1$并验证； 接收$proof_{Paillier,1}$并验证； 接收$proof_{Paillier,2}$并验证； 校验pk长度为$\max\{3\log\|r\|+1,n\}$。
5	计算 Diffie-Hellman **公共密钥** $Q_{common}=x_1\cdot Q_2$，存储(x_1,Q_{common})	计算 Diffie-Hellman **公共密钥** $Q_{common}=x_2\cdot Q_1$，存储 (x_2,Q_{common},c_{key})
	分析：公共私钥$x_{common}=x_1x_1$不出现，双方各自拿 50%。 公共公钥$Q_{common}=x_1x_2\cdot G$。 **Unbound**：公共公钥$Q_{common}=Q_1+Q_2$，公共私钥$x_{common}=x_1+x_2$；	

步骤 1：参与方P_1

1. 选择随机数$x_1\in F_{r/3}$，计算$Q_1=x_1\cdot G$

2. zk-Schnorr 证明 $proof_1 = ZK\left\{x_1 \middle| Q_1 = x_1 \cdot G\right\}$

3. 对 Q_1 和 $proof_1$ 生成承诺与打开承诺 $[KGC_1, KGD_1] = Com(Q_1, proof_1)$

4. 发送承诺 KGC_1 给**理想函数** $\mathcal{F}_{com-zk}^{R_{DL}}$

步骤 2： 参与方 P_2

1. 从理想函数 $\mathcal{F}_{com-zk}^{R_{DL}}$ 接收到承诺 KGC_1

2. 选择随机数 $x_2 \in F_r$，计算 $Q_2 = x_2 \cdot G$

3. zk-Schnorr 证明 $proof_2 = ZK\left\{x_2 \middle| Q_2 = x_2 \cdot G\right\}$

4. 发送 $(proof_2, Q_2)$ 给**理想函数** $\mathcal{F}_{zk}^{R_{DL}}$

步骤 3： 参与方 P_1

1. 从理想函数 $\mathcal{F}_{zk}^{R_{DL}}$ 接收到 $proof_2$，校验，获得 Q_2；

2. 发送打开承诺 KGD_1 给**理想函数** $\mathcal{F}_{com-zk}^{R_{DL}}$

3. 生成 Paillier 密码公钥和私钥 (pk, sk)，长度为 $\max\{3\log|r|+1, n\}$，计算 Paillier 同态加密 $c_{key} := Enc_{pk}(x_1)$，（为签名的同态计算做准备）。

4. zk-Paillier 证明 $proof_{Paillier,1} = ZK\left\{sk \middle| pk = N = p_1 \cdot p_2\right\}$，发送 $proof_{Paillier,1}$ 给**理想函数** $\mathcal{F}_{zk}^{R_{Paillier}}$ （原文证明生成了正确的公钥）

5. 发送 c_{key} 给参与方 P_2

步骤 4：参与方 P_1

1. 发送 zk-Paillier-Enc 证明

$$proof_{Paillier,2} = ZK\left\{(c_{key}, pk, Q_1, G), (x_1, r) \middle| c_{key} = Enc_{pk}(x_1, r), Q_1 = x_1 \cdot G\right\}$$

证明密文 c_{key} 中包含的 x_1 的范围是 F_r，且与 $Q_1 = x_1 \cdot G$ 使用的 x_1 是同一个；

Unbound： zk-Paillier-Enc 证明：1. 密文 $c_{key} = Enc_{pk}(x_1)$ 中 x_1 的范围是 $F_{r/3}$；2.与 $Q_1 = x_1 \cdot G$ 使用的 x_1 是同一个；

步骤 5： 参与方 P_2

1. 从理想函数 $\mathcal{F}_{com-zk}^{R_{DL}}$ 获得打开承诺 KGD_1，从而获得 Q_1 和 $proof_1$，验证有效性
2. 从理想函数 $\mathcal{F}_{zk}^{R_{Paillier}}$ 获得 $proof_{Paillier,1}$，验证有效性
3. 从参与方 P_1 获得 $proof_{Paillier,2}$，验证有效性
4. 校验 pk 长度为 $\max\{3\log|r|+1, n\}$

步骤 6：公共公钥为 Q

1. 参与方 P_1 计算 Diffie-Hellman **公共密钥** $Q_{common} = x_1 \cdot Q_2$，存储 (x_1, Q_{common})；
2. 参与方 P_2 计算 Diffie-Hellman **公共密钥** $Q_{common} = x_2 \cdot Q_1$，存储 $(x_2, Q_{common}, c_{key})$；

注释：公共公钥 $Q_{common} = x_1 x_2 \cdot G$，**公共私钥** $x_{common} = x_1 x_1$ 不出现，双方各自拿 50%。

Unbound： 公共公钥 $Q_{common} = Q_1 + Q_2$，公共私钥 $x_{common} = x_1 + x_2$

3.两方签名

ECDSA 需要计算 R 和 s

- **相同点：** 与上一节的强制诚实版 Diffie-Hellman 密钥交换协议升级版相同（计算 Diffie-Hellman 随机点 R）
- **不同点：** 额外添加 Paillier 密文计算（不泄露任何信息），计算 s

两方需要签名的消息为 $m' = Hash(m) \bmod |F_r|$

	用户 P_1	服务方 P_2
1	①选择随机数 $k_1 \in F_{r/3}$，计算 $R_1 = k_1 \cdot G$ ②zk-Schnorr 证明知道随机数 k_1 $proof_1 = ZK\{k_1 \mid R_1 = k_1 \cdot G\}$ ③对 R_1 和 $proof_1$ 生成承诺与打开承诺 $[KGC_1, KGD_1] = Com(R_1, proof_1)$ **发送承诺** KGC_1	

2		**接收承诺** KGC_1 ①选择随机数 $k_2 \in F_r$，计算 $R_2 = k_2 \cdot G$ ②zk-Schnorr 证明知道随机数 k_2 $proof_2 = ZK\{k_2 \mid R_2 = k_2 \cdot G\}$ **发送** $R_2, proof_2$
3	**接收** $R_2, proof_2$，**校验；** **发送打开承诺** KGD_1	
4		**接收** $R_1, proof_1$，**校验；** ①计算 Diffie-Hellman **公共随机点** $R := k_2 \cdot R_1$，解析 $(r_x, r_y) := R$，计算 $r := r_x \bmod \lvert F_r \rvert$ **（实现 ECDSA 目标 1: R）** ②选择随机数 $\rho \in Z_{F_r^2}$，Paillier 同态加密 $c_1 := Enc_{pk}\left(\rho \cdot \lvert F_r \rvert + \left[k_2^{-1} \cdot m' \bmod \lvert F_r \rvert\right]\right)$ $v := k_2^{-1} \cdot r \cdot x_2 \bmod \lvert F_r \rvert,$ ③同态计算 $c_2 := v \otimes c_{key},$ $c_3 := c_1 \oplus c_2$ 发送 c_3；
	注释： $\rho \cdot \lvert F_r \rvert$ 取值范围是 $Z_{F_r^3}$ 是一个非常大的随机数，对 $\left[k_2^{-1} \cdot m' \bmod \lvert F_r \rvert\right]$ 随机化，且在 Paillier 加密范围内。后面 P_1 解密获得后会模 $\lvert F_r \rvert$ 去掉该项。 $c_{key} := Enc_{pk}(x_1)$ $c_2 := v \otimes c_{key} = Enc_{pk}\left(x_1 \cdot k_2^{-1} \cdot r \cdot x_2 \bmod \lvert F_r \rvert\right), x_{common} = x_1 x_2$ $c_3 := c_1 \oplus c_2 = Enc_{pk}\left(\left(\rho \cdot \lvert F_r \rvert + \left[k_2^{-1} \cdot m' \bmod \lvert F_r \rvert\right]\right) + \left(x_1 \cdot k_2^{-1} \cdot r \cdot x_2 \bmod \lvert F_r \rvert\right)\right)$ $= Enc_{pk}\left(\left(\rho \cdot \lvert F_r \rvert + \left[k_2^{-1} \cdot m' \bmod \lvert F_r \rvert\right]\right) + \left(x_{common} \cdot k_2^{-1} \cdot r \bmod \lvert F_r \rvert\right)\right)$	

<table>
<tr>
<td></td>
<td colspan="2">

取值范围分析：

$$(1) c_1 := Enc_{pk}\left(\rho\cdot|F_r|+\left[k_2^{-1}\cdot m'\bmod|F_r|\right]\right)$$

$$\rho\cdot|F_r|+\left[k_2^{-1}\cdot m'\bmod|F_r|\right]\in F_{r^3}$$

$$(2) v\in|F_r|,$$

$$(3) c_2 := v\otimes c_{key}$$

$$x_1\cdot k_2^{-1}\cdot r\cdot x_2\bmod|F_r|\in|F_{r^2}|$$

$$(4) c_3 := c_1\oplus c_2$$

$$s'=Dec(c_3)=\left(\rho\cdot|F_r|+\left[k_2^{-1}\cdot m'\bmod|F_r|\right]\right)+\left(x\cdot k_2^{-1}\cdot r\bmod|F_r|\right)\in|F_{r^3}|+|F_{r^2}|$$

$$s'<N_{2048bit}$$

密文运算中，对应明文的取值范围远远超过$|F_r|$，**但是不能在密态情况下计算模系数，仅当接收方解密后，才能模系数。**

选择$\rho\in Z_{F_r^2}$的原因：

在$Z_{F_r^2}$范围s'在密态下计算范围恰好在$N_{2048bit}$范围内。

但是，扩大取值范围，如$\rho\in Z_{F_r^3}$，则需要更大范围的$N_{>2048bit}$，降低效率。

$$\rho\cdot|F_r|\in|F_{r^4}|$$

$$c_3 := c_1\oplus c_2$$

$$s'=Dec(c_3)=\left(\rho\cdot|F_r|+\left[k_2^{-1}\cdot m'\bmod|F_r|\right]\right)+\left(x\cdot k_2^{-1}\cdot r\bmod|F_r|\right)\in|F_{r^4}|+|F_{r^2}|$$

$$s'>N_{2048bit}$$

ECC 中的每个倍点运算$H:=x\cdot G$都是严格的离散对数运算，是严格的指数困难，仅需要 256bit 就安全。由于存在大量的合数，排除合数后，大整数因子分解困难问题是亚指数困难，而不是严格的指数困难。

安全性分析：因子分解 2048bit 达到的安全等级与 256bit 的 ECC 相同。因此，这里的公钥pk中的N需要 2048bit。提高N的取值范围，则效率降低。

</td>
</tr>
<tr>
<td>5</td>
<td>

接收c_3；

①计算 Diffie-Hellman **公共随机点** $R:=k_1\cdot R_2$，解析$(r_x,r_y):=R$，计算

$r:=r_x\bmod|F_r|$；

（实现 ECDSA 目标 1: R）

</td>
<td></td>
</tr>
</table>

<table>
<tr><td></td><td>②Paillier 解密 c_3 获得 $s'=Dec_{sk}(c_3)$，
计 算 $s''=k_1^{-1}\cdot s'\bmod|F_r|$ 。 令 $s=\min\{s'',|F_r|-s''\}$；
校验签名 $(m',(r,s),Q)$ 正确性
发送签名 (r,s) 给 P_2（不一定执行）</td><td></td></tr>
<tr><td></td><td colspan="2">注释：
$\rho\cdot|F_r|$ 是 $|F_r|$ 的整数倍，$s''=k_1^{-1}\cdot s'\bmod|F_r|$ 运算模系数后，则去掉这个整数倍的随机项。
$s'=\left(\rho\cdot|F_r|+\left[k_2^{-1}\cdot m'\bmod|F_r|\right]\right)+\left(x_1\cdot k_2^{-1}\cdot r\cdot x_2\bmod|F_r|\right)$
$s''=k_1^{-1}s'\bmod|F_r|=\left[k_1^{-1}k_2^{-1}\cdot m'\bmod|F_r|\right]+\left[x_1\cdot k_1^{-1}k_2^{-1}\cdot r\cdot x_2\bmod|F_r|\right]$
$=\left[k^{-1}m'\bmod|F_r|\right]+\left[k^{-1}x_{common}r\bmod|F_r|\right]$
$=\left[k^{-1}m'+k^{-1}x_{common}r\right]\bmod|F_r|$
$s=\min\{s'',|F_r|-s''\}$
（实现 ECDSA 目标 2： $s:=k^{-1}(m+x_{common}r)=k^{-1}m+k^{-1}x_{common}r$ ）
取值范围分析：
$$s'=Dec_{sk}(c_3)\in|F_{r^3}|+|F_{r^2}|$$
$$s''=k_1^{-1}\cdot s'\bmod|F_r|\in|F_r|$$
密文运算过程中，对应明文的取值范围远远超过 $|F_r|$，但是不能在密态情况下计算模系数。P_1 解密后，进行明文运算是才可以模系数，将范围变小。
分析：两个参与方权利不一样；P_1 权利更大，计算签名（r,s），可以不发送给 P_2。
所以产品方案：P_1 为用户，P_2 为服务方。P_1 生成签名，P_2 提供签名服务。</td></tr>
<tr><td>**6**</td><td></td><td>校验签名 $(m',(r,bool,s),Q_{common})$ 正确性
（不一定执行）</td></tr>
</table>

两方需要签名的消息为 $m'=Hash(m)\bmod|F_r|$

步骤 1： 参与方 P_1

1. 选择一次性随机数$k_1 \in F_{r/3}$，计算$R_1 = k_1 \cdot G$

2. zk-Schnorr 证明 $proof_1 = ZK\{k_1 | R_1 = k_1 \cdot G\}$

3. 对R_1和$proof_1$生成承诺与打开承诺$[KGC_1, KGD_1] = Com(R_1, proof_1)$

4. 发送承诺KGC_1给**理想函数**$\mathcal{F}_{com-zk}^{R_{DL}}$

步骤 2：参与方P_2

1. 从理想函数$\mathcal{F}_{com-zk}^{R_{DL}}$接收到承诺$KGC_1$

2. 选择一次性随机数$k_2 \in F_r$，计算$R_2 = k_2 \cdot G$

3. zk-Schnorr 证明 $proof_2 = ZK\{k_2 | R_2 = k_2 \cdot G\}$

4. 发送$proof_2$、R_2给**理想函数**$\mathcal{F}_{zk}^{R_{DL}}$

步骤 3：参与方P_1

1. 从理想函数$\mathcal{F}_{zk}^{R_{DL}}$接收到$proof_2$、$R_2$，**校验**，获得$R_2$；

2. 发送打开承诺KGD_1给**理想函数**$\mathcal{F}_{com-zk}^{R_{DL}}$

步骤 4：**参与方**P_2

1. 从理想函数$\mathcal{F}_{com-zk}^{R_{DL}}$接收到打开承诺$KGD_1$，获得$R_1$和$proof_1$，**校验**；

2. 计算 Diffie-Hellman **公共随机点**$R := k_2 \cdot R_1$，解析$(r_x, r_y) := R$，计算$r := r_x \bmod |F_r|$

   （实现 ECDSA 目标 1: R）

3. 选择随机数$\rho \in Z_{F_r^2}$，Paillier 同态加密$c_1 := Enc_{pk}\left(\rho \cdot |F_r| + [k_2^{-1} \cdot m' \bmod |F_r|]\right)$，然后计算$v := k_2^{-1} \cdot r \cdot x_2 \bmod |F_r|, c_2 := v \otimes c_{key}, c_3 := c_1 \oplus c_2$ **（密文计算）**

4. 发送c_3给P_1

步骤 5：参与方P_1输出签名

1. 计算 Diffie-Hellman **公共随机点**$R := k_1 \cdot R_2$，解析$(r_x, r_y) := R$，计算$r := r_x \bmod |F_r|$

   （实现 ECDSA 目标 1: R）

2. Paillier 解密 c_3 获得 $s' = Dec_{sk}(c_3)$，计算 $s'' = k_1^{-1} \cdot s' \bmod |F_r|$。令

   $s = \min\{s'', |F_r| - s''\}$。

3. 校验签名 $(m', (r,s), Q_{common})$ 正确性

4. 发送签名 (r,s) 给 P_2（不一定执行）

分析： 两个参与方权利不一样；P_1 权利更大，计算出签名（r, s）后，可以不发送给 P_2。

所以产品会让 P_1 为终端用户，P_2 为服务方。P_1 生成签名，P_2 不需要签名。

步骤 6： 参与方 P_2（不一定执行）

1. 校验签名 $(m', (r, bool, s), Q_{common})$ 正确性

4.分片私钥派生

核心思想： 基于旧分片私钥和旧分片公钥，计算新分片私钥和新分片公钥，而不需要重新运行**密钥生成协议**。
旧分片私钥和旧分片公钥作为**预计算**。

初始状态：

1. 用户 P_1 的公钥为 Q_1，私钥为 x_1；
2. 服务方 P_2 的公钥为 Q_2，私钥为 x_2；
3. 双方的公共公钥为 $Q_{common} = x_1 x_2 \cdot G = x_{common} \cdot G$；

步骤 1： 双方运行强制诚实版的 Diffie-Hellman 密钥交换协议的升级版：

双方获得 Diffie-Hellman 会话密钥为 $\tilde{Q}_{common}$，然后计算哈希值，称为：链码 chain code

$$cc = sha256(\tilde{Q}_{common})$$

该链码用于密钥更新。

步骤 2： 双方均计算

对于一个给定的公开计数器 $counter$，双方均计算：

$$f = HMAC512(cc, Q_{common}, counter)$$
$$f = f_l \| f_r, where |f_l| = |f_r| = 256$$
$$Q_{1,new} := f_l \cdot Q_1$$
$$Q_{common,new} := f_l \cdot Q_{common}$$
$$cc_{new} = f_r \cdot cc$$

计算后产生的结果：

1. 用户 P_1 分片私钥派生为 $x_{1,new} = f_l \cdot x_1$，分片公钥派生为 $Q_{1,new} = f_l \cdot Q_2$
2. 服务方 P_2 分片私钥和分片公钥均**不变**；
3. **公共公钥**派生为 $Q_{common,new} := f_l \cdot Q_{common}$；公共私钥也派生，但是不出现。

公式推导：

$$Q_{common,new} := f_l \cdot Q_{common} = (f_l \cdot x_1)x_2 \cdot G = x_{1,new}x_2 \cdot G$$

$cc_{new} = f_r \cdot cc$ 为新链码 chain code，不再需要运行强制诚实版的 Diffie-Hellman 密钥交换协议升级版。

5.分片私钥刷新

分片私钥刷新，但公共私钥不变、公共公钥不变

$$9_{common} = 6+3 = 5+4 = 7+2 = 8+1$$
$$x_{common} = x_1 x_2 = (r^{-1}x_1)(rx_2)$$
$$x_{common} = x_1 + x_2 = (x_1 + \Delta) + (x_2 - \Delta)$$

Step1：双方进行**掷币协议**

初始化：椭圆曲线生成元为 G，标量域为 F_r，基域为 F_q。

P_1	P_2
选择随机数 $r_1 \in F_r$，计算承诺与打开承诺 $[C, D] := Com(r_1)$，发送承诺 C；	
	选择一个随机数 $r_2 \in F_r$，发送 r_2；
计算 $r := r_1 \oplus r_2 \in F_r$，发送打开承诺 D；	
	校验承诺 $Verify(C, D)$ 后，计算 $r := r_1 \oplus r_2 \in F_r$

运行结果：双方获得**公共随机数** $r \in F_r$。 分析：使用 Diffie-Hellman 协议效果一样，但是需要 zk-Schnorr，计算复杂度更高。

Step2：双方分片私钥刷新，公共公钥和公共私钥不变

P_1	P_2
分片私钥和分片公钥更新为 $$x_{1,new} := r^{-1}x_1, Q_{1,new} = x_{1,new} \cdot G$$ 生成新的 Paillier 密钥对 (pk_{new}, sk_{new})，用于签名过程中的同态计算。 ①zk-Schnoor 证明知道 ECC 新的分片私钥 $$proof_1 = ZK\left\{x_{1,new} \middle\| Q_{1,new} = x_{1,new} \cdot G\right\}$$ ②zk-Paillier-N 证明知道 Paillier 新的私钥 $$proof_{Paillier,1} = ZK\left\{sk \middle\| pk = N = p_1 \cdot p_2\right\}$$ ； ③zk-Paillier-Enc 正确加密 ECC 私钥且 ECC 私钥范围正确 $$proof_{Paillier,2} = ZK\left\{c_{key}, pk, Q_1' \in L_{PDL}\right\}$$ 发送 $proof_1, proof_{Paillier,1}, proof_{Paillier,2}$；	
	接收 $proof_1, proof_{Paillier,1}, proof_{Paillier,2}$ 并**校验；** 分片私钥和分片公钥刷新为 $$x_{2,new} := rx_2, Q_{2,new} = x_{2,new} \cdot G$$ zk-Paillier-N 证明知道 ECC 分片私钥 $$proof_2 = ZK\left\{x_{2,new} \middle\| Q_{2,new} = x_{2,new} \cdot G\right\}$$； 发送 $proof_2$。
接收 $proof_2$ 并校验。	
分析： 用户 P_1 私钥和公钥刷新为 $x_{1,new} := r^{-1}x_1, Q_{1,new} = x_{1,new} \cdot G$ 用户 P_2 私钥和公钥刷新为 $x_{2,new} := rx_2, Q_{2,new} = x_{2,new} \cdot G$ **公共公钥不变：**	

$$Q_{new}=x_{1,new}\cdot Q_{2,new}=x_{2,new}\cdot Q_{1,new}=x_{1,new}x_{2,new}\cdot G=(r^{-1}x_1)(rx_2)\cdot G=x_1x_2\cdot G=Q_{old}$$

Unbound 差异：

由于：公共公钥 $Q_{common}=Q_1+Q_2$，公共私钥 $x_{common}=x_1+x_2$

用户 P_1 分片私钥和分片公钥刷新为 $x_{1,new}:=x_1-r,Q_{1,new}=(x_{1,new}-r)\cdot G$

用户 P_2 分片私钥和分片公钥刷新为 $x_{2,new}:=(x_2+r),Q_{2,new}=(x_{2,new}+r)\cdot G$

$$x_{common}=x_{1,new}+x_{2,new}=x_1+x_2,$$
$$Q_{new}=Q_{1,new}+Q_{2,new}=(x_1+x_2)\cdot G=Q_{common}$$

刷新分片私钥，公共公钥不变。

6.恢复公共私钥

应用需求：服务方不 work，用户恢复公共私钥。

托管方的私钥和公钥分别为 $(k_e,Q_e),Q_e=k_e\cdot G$ 。

步骤 1：服务方 P_2 将私钥 x_2 分为 n 个片段 $\{x_{2,1},...,x_{2,n}\}$ 使得离散对数计算**不再困难，暴力搜索时间很短**。使用**托管方**的公钥 Q_e 对 n 个片段分别进行 ElGamal 加密

$$(D_i,E_i):=\left(x_{2,i}\cdot G+r_i\cdot Q_e,r_i\cdot G\right)_{i\in\{1,...,n\}}$$

将密文发送给 P_1。

步骤 2：服务方 P_2 发送三个 zk 给用户 P_1

$$ZK\left\{x_{2,i}\middle|x_2=\sum_{i=1}^{n}x_{2,i}\right\}_{i\in\{1,...,n\}}$$
$$ZK\left\{x_{2,i},r_i\middle|\begin{pmatrix}D_i=x_{2,i}\cdot G+r_i\cdot Q_e,\\E_i=r_i\cdot G\end{pmatrix}_{i\in\{1,...,n\}}\right\}$$
$$Bullet\_proofs\left\{x_{2,i}\middle|0\le x_{2,i}<K\right\}_{i\in\{1,...,n\}}$$

其中，K 为一个范围较小的常量。

步骤 3：托管方周期性使用私钥 k_e 对当天新闻签名并发送给用户；用户验证签名；表明托管方一直拥有正确的私钥 k_e 。

步骤 4： 如果**托管方**检测到服务方 P_2 长时间不工作，则托管方公开私钥 k_e，则用户 P_1 获得 k_e 后，计算

$$x_{2,i}\cdot G := \left(D_i - k_e \cdot E_i\right)_{i\in\{1,\ldots,n\}}$$

并暴力搜索 $x_{2,i} \leftarrow \{x_{2,i}\cdot G\}_{i=1,\ldots,n}$，然后计算服务方的私钥 $x_2 := \sum_{i=1}^{n} x_{2,i}$，然后恢复完整私钥 $x = x_1 x_2$。

KZen T. Bitcoin Wallet Powered by Two-Party ECDSA-Extended Abstract[J]. Retrieved March, 2021.

lynndell 新火科技 密码学专家 lynndell2010@gmail.com