密码学原理与实践

（第三版）

第6章 公钥密码学和离散对数

苏 明

[加] Douglas R. Stinson 著

冯登国 等译

概览

- 6．1 ElGamal密码体制
- 6．2 离散对数问题的算法
- 6．3 通用算法的复杂度下界
- 6．4 有限域
- 6．5 椭圆曲线
- 6．6 实际中的离散对数算法
- 6．7 ElGamal体制的安全性

离散对数问题

离散对数

实例： 乘法群 $(G,\cdot)$，一个 n 阶元素 $\alpha \in G$ 和元素 $\beta \in \langle\alpha\rangle$

问题： 找到唯一的整数 a，$0 \leqslant a \leqslant n-1$，满足

$$\alpha^a = \beta$$

我们将这个整数 a 记为 $\log_\alpha \beta$，称为 β 的离散对数。

6.2 离散对数问题的算法

Shanks算法 (时间-存储折中算法)

算法 SHANKS(G, n, α, β)

1. $m \leftarrow \lceil \sqrt{n} \rceil$
2. **for** $j \leftarrow 0$ **to** $m-1$
   do 计算 α^{mj}
3. 对 m 个有序对 (j, α^{mj}) 关于第二个坐标排序，得到一个列表 L_1
4. **for** $i \leftarrow 0$ **to** $m-1$
   do 计算 $\beta\alpha^{-i}$
5. 对 m 个有序对 $(i, \beta\alpha^{-i})$ 关于第二个坐标排序，得到一个列表 L_2
6. 找到对 $(j, y) \in L_1$ 和 $(i, y) \in L_2$ （即找到两个具有相同第二坐标的对）
7. $\log_\alpha \beta \leftarrow (mj+i) \bmod n$

6.2 离散对数问题的算法

假定我们要在 $(\mathbb{Z}_{809}^*, \cdot)$ 中求出 $\log_3 525$。注意 809 是素数，3 是 $\mathbb{Z}_{809}^*$ 中本原元，这时 $\alpha=3$，$n=808$，$\beta=525$ 和 $m=\lceil\sqrt{808}\rceil=29$。则

$$\alpha^{29} \bmod 809 = 99$$

首先，对于 $0 \leqslant j \leqslant 28$ 计算有序对 $(j, 99^j \bmod 809)$。得到下面的列表

(0, 1)	(1, 99)	(2, 93)	(3, 308)	(4, 559)
(5, 329)	(6, 211)	(7, 664)	(8, 207)	(9, 268)
(10, 644)	(11, 654)	(12, 26)	(13, 147)	(14, 800)
(15, 727)	(16, 781)	(17, 464)	(18, 632)	(19, 275)
(20, 528)	(21, 496)	(22, 564)	(23, 15)	(24, 676)
(25, 586)	(26, 575)	(27, 295)	(28, 81)	

这些序对排序后产生 L_1。

$$\log_3 525 = (29\times 10+19) \bmod 809 = 309$$

第二个列表包括序对 $(i, 525\times(3^i)^{-1} \bmod 809)$，$0 \leqslant i \leqslant 28$。如下所示：

(0, 525)	(1, 175)	(2, 328)	(3, 379)	(4, 396)
(5, 132)	(6, 44)	(7, 554)	(8, 724)	(9, 511)
(10, 440)	(11, 686)	(12, 768)	(13, 256)	(14, 355)
(15, 388)	(16, 399)	(17, 133)	(18, 314)	(19, 644)
(20, 754)	(21, 521)	(22, 713)	(23, 777)	(24, 259)
(25, 356)	(26, 658)	(27, 489)	(28, 163)	

排序后得到 L_2。

6.2 离散对数问题的算法

- Shanks算法的时间、空间复杂度？

- Storage Cost: $O(\sqrt{n})$

- Time Cost: $O(\sqrt{n})+O(mlogm)= O(\sqrt{n}logn)$

6.3 通用算法的复杂度下界

- Pollard Rho 离散对数算法
- Pohlig-Hellman算法
- 指数演算法

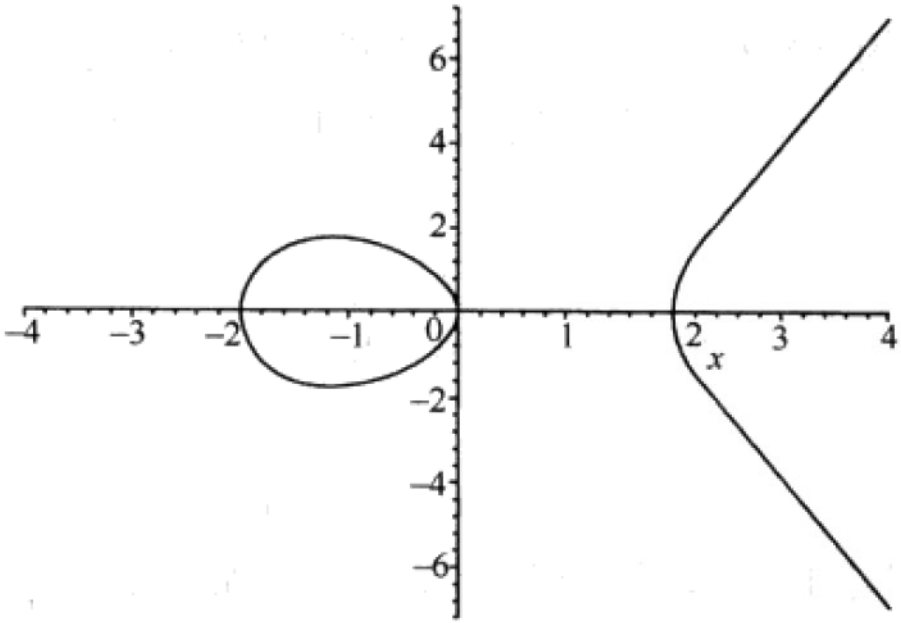
离散对数问题通用算法的复杂度：$\Omega(\sqrt{n})$

6.5 椭圆曲线

定义 设 $a, b \in \mathbb{R}$ 是满足 $4a^3 + 27b^2 \neq 0$ 的常实数。方程

$$y^2 = x^3 + ax + b$$

的所有解 $(x, y) \in \mathbb{R} \times \mathbb{R}$ 连同一个无穷远点 O 组成的集合 E 称为一个非奇异椭圆曲线。

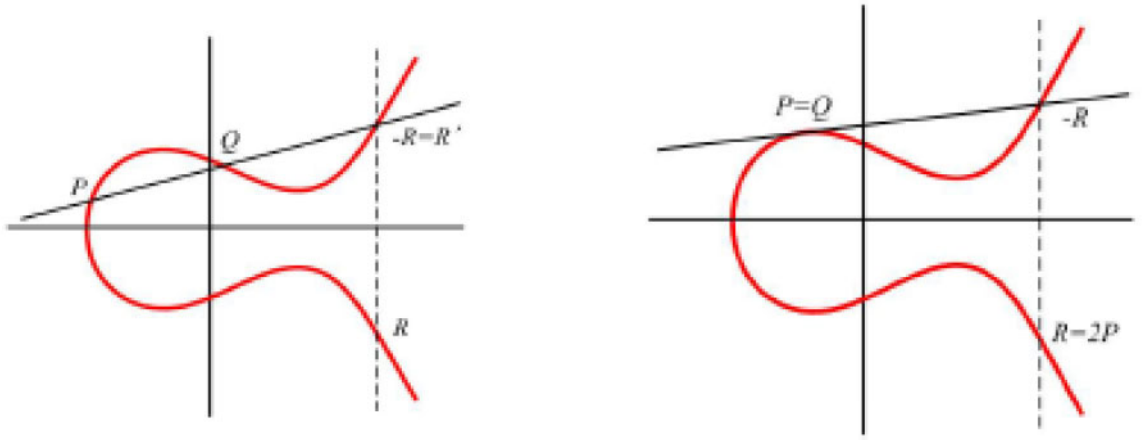


$y^2 = x^3 - 4x$

Elliptic Curve Cryptosystem

$$y^2 = x^3 + ax + b$$

$(E, +, \mathcal{O})$ is an abelian group where the infinity point is the identity in the group.



(a) (b)

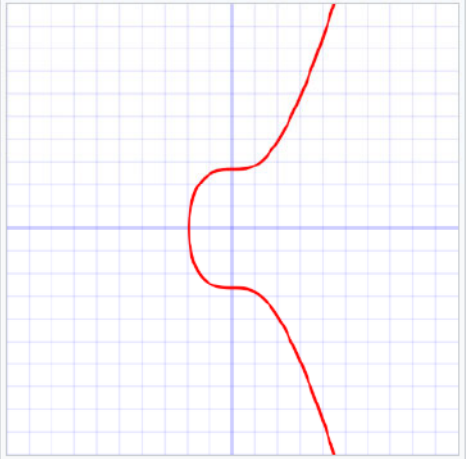
Point addition and point doubling

Secp256k1

Secp256k1

secp256k1 refers to the parameters of the elliptic curve used in Bitcoin's public-key cryptography, and is defined in *Standards for Efficient Cryptography (SEC)* (Certicom Research, http://www.secg.org/sec2-v2.pdf). Currently Bitcoin uses secp256k1 with the ECDSA algorithm, though the same curve with the same public/private keys can be used in some other algorithms such as Schnorr.

secp256k1 was almost never used before Bitcoin became popular, but it is now gaining in popularity due to its several nice properties. Most commonly-used curves have a random structure, but secp256k1 was constructed in a special non-random way which allows for especially efficient computation. As a result, it is often more than 30% faster than other curves if the implementation is sufficiently optimized. Also, unlike the popular NIST curves, secp256k1's constants were selected in a predictable way, which significantly reduces the possibility that the curve's creator inserted any sort of backdoor into the curve.



This is a graph of secp256k1's elliptic curve $y^2 = x^3 + 7$ over the real numbers. Note that because secp256k1 is actually defined over the field Z_p, its graph will in reality look like random scattered points, not anything like this.

Technical details

As excerpted from *Standards*:

The elliptic curve domain parameters over F_p associated with a Koblitz curve secp256k1 are specified by the sextuple T = (p,a,b,G,n,h) where the finite field F_p is defined by:

- p = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFC2F
- = $2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$

The curve E: $y^2 = x^3+ax+b$ over F_p is defined by:

- a = 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
- b = 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000007

The base point G in compressed form is:

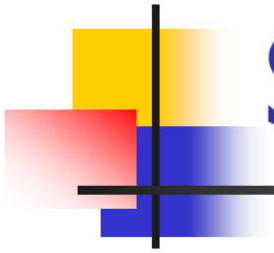
- G = 02 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798

and in uncompressed form is:

- G = 04 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9 59F2815B 16F81798 483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419 9C47D08F FB10D4B8

Finally the order n of G and the cofactor are:

- n = FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE BAAEDCE6 AF48A03B BFD25E8C D0364141
- h = 01



SM2

SM2椭圆曲线公钥密码算法推荐曲线参数

推荐使用素数域256位椭圆曲线。

椭圆曲线方程：$y^2 = x^3 + ax + b$。

曲线参数：

p=FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF

a=FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFC

b=28E9FA9E 9D9F5E34 4D5A9E4B CF6509A7 F39789F5 15AB8F92 DDBCBD41 4D940E93

n=FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF 7203DF6B 21C6052B 53BBF409 39D54123

G_x=32C4AE2C 1F198119 5F990446 6A39C994 8FE30BBF F2660BE1 715A4589 334C74C7

G_y=BC3736A2 F4F6779C 59BDCEE3 6B692153 D0A9877C C62A4740 02DF32E5 2139F0A0

Elliptic Curve Cryptosystem

- Addition Formulas: Nonbinary case, i.e., F = GF(p); p > 3.
- For P = $(x_1;y_1)$ and Q =$(x_2;y_2)$, then P + Q = R = $(x_3;y_3)$ for Q !=-P

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \end{aligned} \qquad \text{where } \lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q \\ \frac{3x_1^2 + a}{2y_1} & \text{if } P = Q \end{cases}$$

6.5 椭圆曲线

- (E,+)构成一个加法群 (扩展到Z_p上)

1. 加法在集合 E 上是封闭的。
2. 加法是可交换的。
3. O 是加法的单位元。
4. E 上每个点有关于加法的逆元。

直观：椭圆曲线加法群+运算比RSA*运算更复杂；
相同的安全强度： 1024 bit RSA ~ 167 ECC

6.5 椭圆曲线

例子 设 E 是 $\mathbb{Z}_{11}$ 上的椭圆曲线 $y^2 = x^3 + x + 6$

x	$x^3+x+6 \bmod 11$	是二次剩余吗	y
0	6	否	
1	8	否	
2	5	是	4, 7
3	3	是	5, 6
4	8	否	
5	4	是	2, 9
6	8	否	
7	4	是	2, 9
8	9	是	3, 8
9	7	否	
10	4	是	2, 9

数乘：$k\alpha = \alpha + \cdots + \alpha$ (k个)

请计算 $2\alpha = 2 \cdot (2,7) = ?$

6.5 椭圆曲线

- Hasse定理： $p+1-2\sqrt{p} \leqslant \#E \leqslant p+1+2\sqrt{p}$

- 点压缩： Point-Compress: $E \setminus \{\mathcal{O}\} \rightarrow \mathbb{Z}_p \times \mathbb{Z}_2$

算法 Point-Decompress(x, i)

$z \leftarrow x^3 + ax + b \bmod p$

if z 是模 p 的非二次剩余类

then return(“failure”)

else
- $y \leftarrow \sqrt{z} \bmod p$
- **if** $y \equiv i \pmod 2$
  - **then return** (x, y)
  - **else return** $(x, p-y)$

6.5 椭圆曲线上的数乘

Computing Point Multiples on Elliptic Curves

- **Double & Add** (Traditional)
- **NAF**表示
- 整数c的一个带符号的二进制表示：$(c_{\ell-1}, \ldots, c_0)$
  没有两个连续的c_i非零

例如：(0, 1, … , 1, 1) ⟶ (1, 0, …,0, -1)

6.5 椭圆曲线

算法 倍数-和差算法 Double-And-(Add Or Subtract) $(P,(c_{\ell-1},\cdots,c_0))$

$Q \leftarrow \mathcal{O}$

for $i \leftarrow \ell-1$ **downto** 0

do
- $Q \leftarrow 2Q$
- **if** $c_i = 1$
  - **then** $Q \leftarrow Q+P$
- **else if** $c_i = -1$
  - **then** $Q \leftarrow Q-P$

return(Q)

11% Speedup VS Traditional Method

6.7 ElGamal密码体制

密码体制　　$\mathbb{Z}_p^*$ 上的 ElGamal 公钥密码体制

设 p 是一个素数，使得 $(\mathbb{Z}_p^*, \cdot)$ 上的离散对数问题是难处理的，令 $\alpha \in \mathbb{Z}_p^*$ 是一个本原元。令 $\mathcal{P} = \mathbb{Z}_p^*$，$\mathcal{C} = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$，定义

$$\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}$$

p, α, β 是公钥，a 是私钥。

对 $K = (p, \alpha, a, \beta)$，以及一个（秘密）随机数 $k \in \mathbb{Z}_{p-1}$，定义

$$e_K(x, k) = (y_1, y_2)$$

其中

$$y_1 = \alpha^k \bmod p$$

且

$$y_2 = x\beta^k \bmod p$$

对 $y_1, y_2 \in \mathbb{Z}_p^*$，定义

$$d_K(y_1, y_2) = y_2 (y_1^a)^{-1} \bmod p$$

一个明文可能会有多个密文！

6.7 ElGamal密码体制

Extension to EC ElGamal

密码体制

令 E 是定义在 $\mathbb{Z}_p$（$p>3$ 是素数）上的一个椭圆曲线，E 包含一个素数阶 n 的循环子群 $H=\langle P\rangle$，其上的离散对数问题是难解的。

设 $\mathcal{P}=\mathbb{Z}_p^*$，$\mathcal{C}=(\mathbb{Z}_p\times\mathbb{Z}_2)\times\mathbb{Z}_p^*$，定义

$$\mathcal{K}=\{(E,P,m,Q,n):Q=mP\}$$

值 P,Q 和 n 是公钥，$m\in\mathbb{Z}_n^*$ 是私钥。

对 $K=(E,P,m,Q,n)$，一个（秘密）随机数 $k\in\mathbb{Z}_n^*$，以及 $x\in\mathbb{Z}_p^*$，定义

$$e_K(x,k)=(\text{Point-Compress}(kP),\ xx_0 \bmod p)$$

其中 $kQ=(x_0,y_0)$ 和 $x_0\neq 0$。

对密文 $y=(y_1,y_2)$，这里 $y_1\in\mathbb{Z}_p\times\mathbb{Z}_2$ 和 $y_2\in\mathbb{Z}_p^*$，定义

$$d_K(y)=y_2(x_0)^{-1} \bmod p$$

其中

$$(x_0,y_0)=m\text{Point-Decompress}(y_1)$$

6.7 ElGamal体制的安全性

- 离散对数的比特安全性

β 是二次剩余当且仅当 $\log_\alpha \beta$ 是偶数

$$L_1(\beta)=\begin{cases}0 & \beta^{(p-1)/2}\equiv 1(\bmod p)\\ 1 & \text{其他}\end{cases}$$

6.7 ElGamal体制的安全性

算法　L_2 Oracle-Discrete-Logarithm (p, α, β)

external L_1, **Oracle** L_2

$x_0 \leftarrow L_1(\beta)$

$\beta \leftarrow \beta / \alpha^{x_0} \bmod p$

$i \leftarrow 1$

while $\beta \neq 1$

do
- $x_i \leftarrow$ Oracle $L_2(\beta)$
- $\gamma \leftarrow \beta^{(p+1)/4} \bmod p$
- **if** $L_1(\gamma) = x_i$
  - **then** $\beta \leftarrow \gamma$
  - **else** $\beta \leftarrow p - \gamma$
- $\beta \leftarrow \beta / \alpha^{x_i} \bmod p$
- $i \leftarrow i + 1$

return $(x_{i-1}, x_{i-2}, \cdots, x_0)$

$$\log_\alpha \beta = \sum_{i \geqslant 0} x_i 2^i$$

6.7 ElGamal体制的安全性

- 关键点：

若β是Zp中的二次剩余，也就是$\beta = \alpha^{a}$(a even)

$$L_2(\beta) = L_1(\alpha^{\frac{a}{2}})$$

若β是Zp中的二次非剩余，那么

$\beta = \frac{\beta}{\alpha^{L_1(\beta)}}$ 就是二次剩余

6.7 ElGamal体制的安全性

- 语义安全性

密码体制 $\mathbb{Z}_p^*$ 上的 ElGamal 公钥密码体制

设 p 是一个素数，使得 $(\mathbb{Z}_p^*, \cdot)$ 上的离散对数问题是难处理的，令 $\alpha \in \mathbb{Z}_p^*$ 是一个本原元。令 $\mathcal{P} = \mathbb{Z}_p^*$，$\mathcal{C} = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$，定义

$$\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}$$

p, α, β 是公钥，a 是私钥。

对 $K = (p, \alpha, a, \beta)$，以及一个（秘密）随机数 $k \in \mathbb{Z}_{p-1}$，定义

$$e_K(x, k) = (y_1, y_2)$$

其中

$$y_1 = \alpha^k \bmod p$$

且

$$y_2 = x\beta^k \bmod p$$

对 $y_1, y_2 \in \mathbb{Z}_p^*$，定义

$$d_K(y_1, y_2) = y_2 (y_1^a)^{-1} \bmod p$$

该密码体制**不是语义安全的**： 因为可以通过二次剩余判定a,k的奇偶性

因此仅从密文可以区分明文(x_1， x_2：若它们二次剩余属性不同)

6.7 ElGamal体制的安全性

- Diffie-Hellman问题

问题 6.3　Computational Diffie-Hellman

实例：一个乘法群 $(G,\cdot)$，一个 n 阶元素 $\alpha \in G$，两个元素 $\beta, \gamma \in \langle \alpha \rangle$。

问题：找出 $\delta \in \langle \alpha \rangle$，满足 $\log_\alpha \delta \equiv \log_\alpha \beta \times \log_\alpha \gamma (\mathrm{mod}\, n)$（等价地，给定 α^b 和 α^c，找出 α^{bc}）。

问题 6.4　Decision Diffie-Hellman

实例：一个乘法群 $(G,\cdot)$，一个 n 阶元素 $\alpha \in G$，三个元素 $\beta, \gamma, \delta \in \langle \alpha \rangle$。

问题：是否有 $\log_\alpha \delta \equiv \log_\alpha \beta \times \log_\alpha \gamma (\mathrm{mod}\, n)$？[等价地，给定 α^b，α^c 和 α^d，判定是否 $d \equiv bc (\mathrm{mod}\, n)$]。

6.7 ElGamal体制的安全性

- 可以知道

$$\mathrm{DDH} \propto_T \mathrm{CDH}$$

the assumption that **DDH** is infeasible is at least as strong as the assumption that **CDH** is infeasible

$$\mathrm{CDH} \propto_T \text{Discrete Logarithm}$$

6.7 ElGamal体制的安全性

- ElGamal decryption (without knowing the private key) $\equiv_T$ solving **CDH**

假设 OracleCDH 是解 CDH 的一个算法，设 (y_1, y_2) 是 ElGamal 密码体制的密文，具有公钥 α 和 β。计算

$$\delta = \text{OracleCDH}(\alpha, \beta, y_1)$$

然后定义

$$x = y_2\delta^{-1}$$

容易看出，x 是密文 (y_1, y_2) 的解密。

定义 α 和 β 是 ElGamal 密码体制的公钥。那么，定义 $y_1 = \gamma$，令 $y_2 \in \langle\alpha\rangle$ 为随机取定。计算

$$x = \text{Oracle-ElGamal-Decrypt}(\alpha, \beta, (y_1, y_2))$$

这是密文 (y_1, y_2) 的解密。最后，计算

$$\delta = y_2x^{-1}$$

δ 是 CDH 给定实例的解。