密码学实验实验报告九

18374480-黄翔

2021 年 6 月 8 日

1 实验目的

1. 掌握椭圆曲线上的运算和常见的椭圆曲线密码算法
2. 了解基于 ECC 的伪随机数生成算法和基于椭圆曲线的商用密码算法

2 实验环境

1. python 3.9

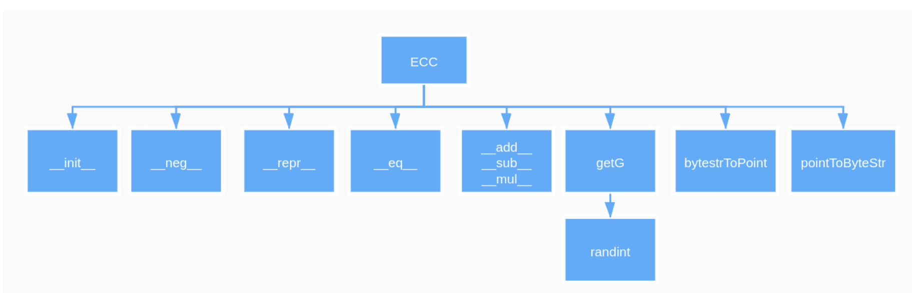
3 实验内容

3.1 ECC 四则运算与消息编码

3.1.1 算法流程图

减法只需要取逆 (y 取负) 即可

3.1.2 函数调用关系



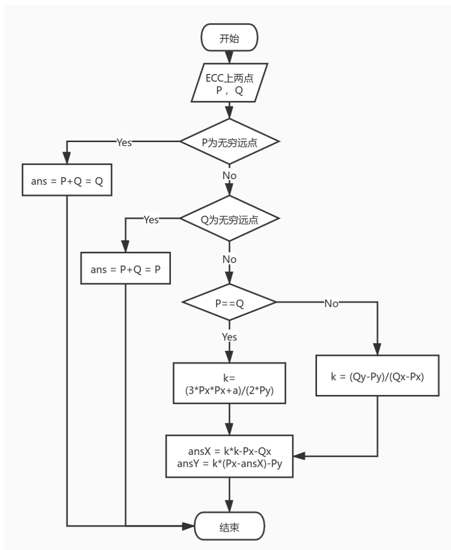


图 1: 加法

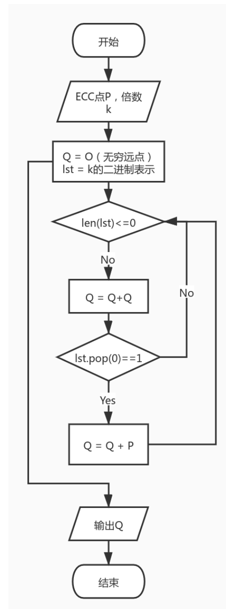


图 2: 乘法

3.1.3 算法伪代码

Algorithm 1 ECC 加法

Input: P, Q

Output: $ans = P + Q$

```
O 为 ECC 无穷远点
if P is O then
    return ans ← Q
end if
if Q is O then
    return ans ← P
end if
if P == Q then
    k ← (3×P_x^2+a)/(2×P_y)
else
    k ← (Q_y−P_y)/(Q_x−P_x)
end if
```

$$k \leftarrow \frac{3\times P_x^2+a}{2\times P_y}$$

$$k \leftarrow \frac{Q_y-P_y}{Q_x-P_x}$$

$ans.X = k^2 - P_x - Q_x$

$ans.Y = k \times (P_x - ans.X) - P_y$

return ans

Algorithm 2 ECC 减法

Input: P, Q

Output: ans

$-Q = (Q_x, -Q_y)$

return $ans \leftarrow P + (-Q)$

Algorithm 3 ECC 乘法

Input: P, k

Output: $Q = [k]P$

$list \leftarrow k$ 的二进制表示

$Q \leftarrow O$（无穷远点）

while $len(list) > 0$ **do**

  $Q \leftarrow Q + Q$

  if $list.pop(0)$ is 1 **then**

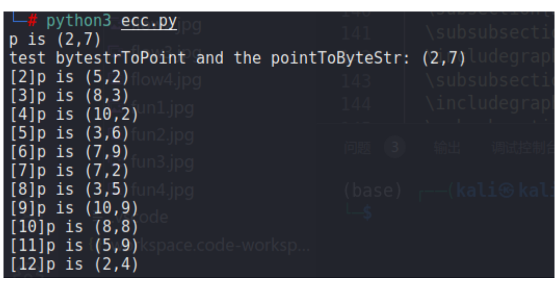
    $Q \leftarrow Q + P$

  end if

end while

return Q

3.1.4 测试样例及结果截图



3.1.5 总结

点算法复杂度分析 I, S, M 分别表示有限域上的求逆运算、乘方以及乘法运算。一般的，有 $I = 6 \times M, S = 0.8 \times M$

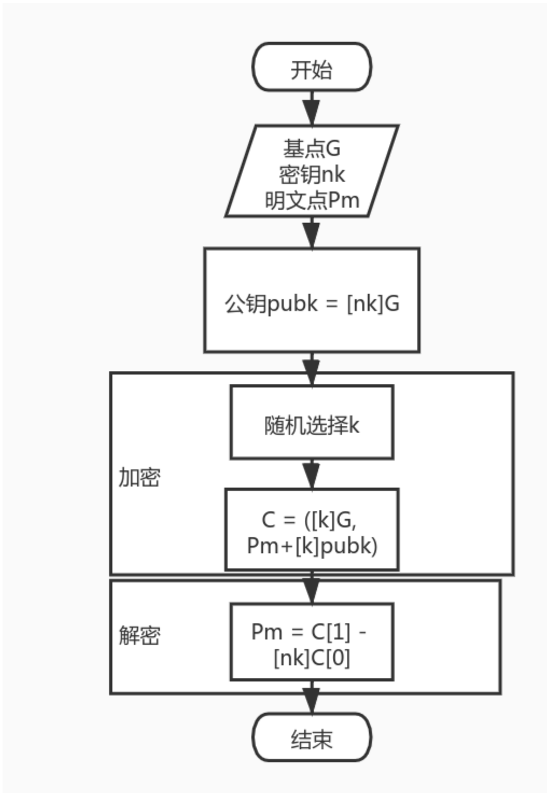
运算	计算量
$P+Q$	$I+2M+S$
$2P$	$I+2M+2S$
$[k]P($ 二进制优化算法)	$1.5l+3lM+2.5lS(l$ 为 k 比特数)

表 1: ECC 点算法时间复杂度

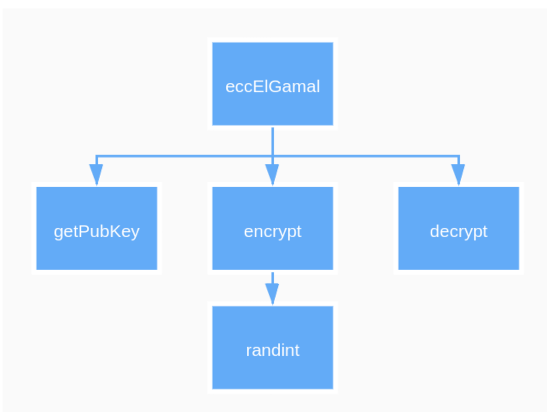
倍点运算优化思路 从 k 转换为二进制计算的优化来看，主要是运用我们有现成的公式计算 $[2]P$，实际上，我们可以找到 $[3]P$,$[5]P$,$[7]P$ 的计算方法，因此类似的，我们可以将 k 分解 $2,3$ 等基底的表示。这就是 ECC 多基数链加速的方法。

3.2 ECC 的 ElGamal 加解密

3.2.1 算法流程图



3.2.2 函数调用关系



3.2.3 算法伪代码

Algorithm 4 eccElGamal 加密

Input: 明文点 P_m，公钥 Pub_k，基点 G

Output: 密文组 C_m

1: $k \leftarrow$ 随机选择正整数

2: $C_m \leftarrow ([k]G, P_m + [k]Pub_k)$

3: **return** C_m

Algorithm 5 eccElGamal 加密

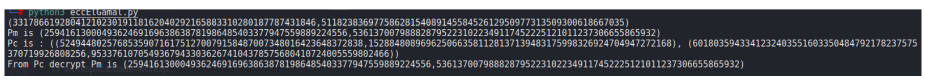
Input: 密文组 C_m，私钥 n_k，基点 G

Output: 明文点 P_m

1: $P_m \leftarrow C_m[1] - n_k C_m[0]$

2: **return** P_m

3.2.4 测试样例及结果截图

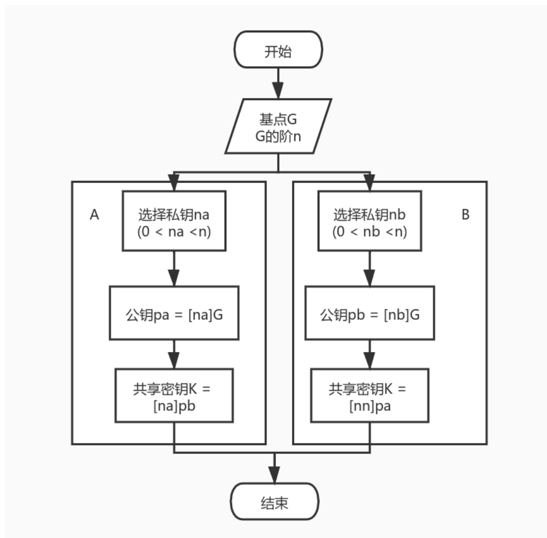


3.2.5 总结

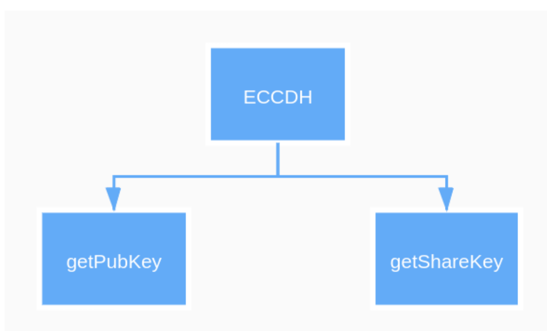
见 3.3.5 ECCDH 部分

3.3 ECCDH

3.3.1 算法流程图



3.3.2 函数调用关系



3.3.3 算法伪代码

Algorithm 6 ECCDH

Input: 基点 G，阶数 n

Output: 共享密钥 K

1: A 选择私钥 n_A，B 选择私钥 n_B

2: A 生成公钥 $P_A \leftarrow [n_A]G$，B 生成公钥 $P_B \leftarrow [n_B]G$

3: A 计算共享密钥 $K \leftarrow [n_A]P_B$，B 计算共享密钥 $K \leftarrow [n_B]P_A$

return K

3.3.4 测试样例及结果截图

```
└─# python3 eccDH.py
The public Key of A: (115,48)
The public Key of B: (130,203)
The shareKey A gets is (161,69)
The shareKey B gets is (161,69)
```

3.3.5 总结

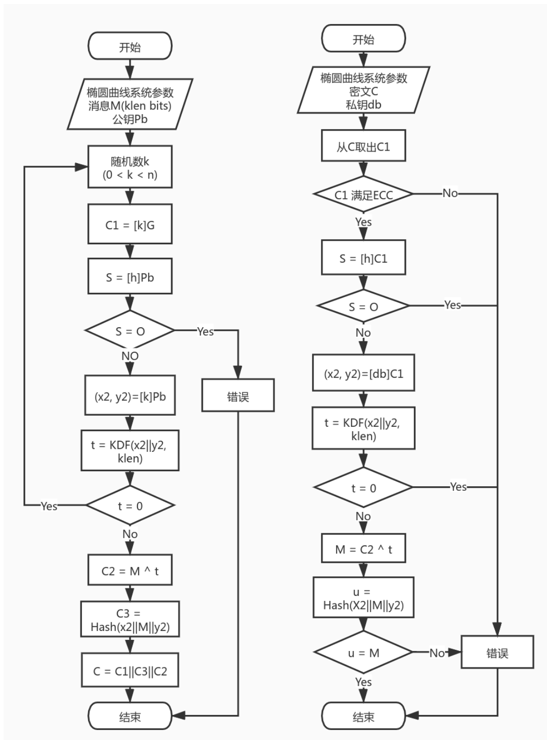
ECC 密码体制安全性 对于一般椭圆曲线的离散对数问题，目前只存在指数级计算复杂度的求解方法。ECC 密码体制的安全性取决于椭圆曲线的离散对数问题。即对 $Q=[k]P$，$Q,P\in E_p(a,b)$ 且 $k<p$，给定 k 与 P 计算 Q 是容易的，但是给定 P 和 Q 计算 k 困难

ECC 密码体制常见攻击方法 ECC 基点 G 的阶为 N

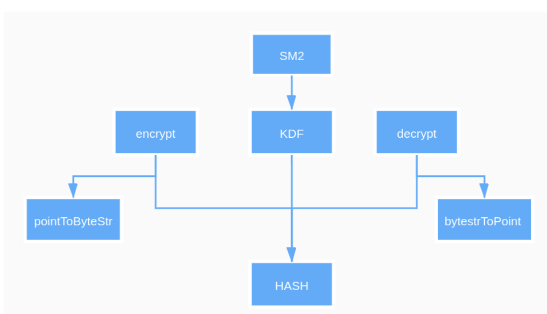
1. 穷举搜索法：计算 $[2]P,[3]P$ 直到找到 $k, Q=[k]P$，复杂度为 $O(N)$
2. 大步小步 (BSGS) 算法。时间复杂度为 $O(\sqrt{N})$
3. SPH 攻击法：该算法实质上是一种演化算法, 它的最大功能是将阶为 N 的群上的离散对数问题化为阶为 N 的一个素因子的循环子群上的离散对数问题。
4. MOV 攻击。只对超奇异椭圆曲线曲线
5. Weil descent 攻击。只适合于特征值为 2 的复合域
6. Pollard's rho 算法攻击。

3.4 sm2 加解密

3.4.1 算法流程图



3.4.2 函数调用关系



3.4.3 算法伪代码

Algorithm 7 sm2 加密

Input: 基点 G，阶数 n，明文 M，公钥 Pub_k

Output: 密文 $C = C_1||C_3||C_2$

1: $klen \leftarrow len(M) * 8$
2: $n \leftarrow G$ 的阶
3: $k \leftarrow randint(1, n-1)$
4: $C_1 \leftarrow pointToByteStr([k]G)$
5: $S \leftarrow [h]Pub_k$
6: **if** S is O **then**
7:     **return** $Wrong$
8: **end if**
9: $(x_2, y_2) \leftarrow [k]Pub_k$
10: $t \leftarrow KDF(x_2||y_2, klen)$
11: **if** t is 0 **then**
12:     **goto** $line3$
13: **end if**
14: $C_2 \leftarrow M \oplus t$
15: $C_3 \leftarrow Hash(x_2||M||y_2)$
16: $C \leftarrow C_1||C_3||C_2$
17: **return** C

Algorithm 8 sm2 解密

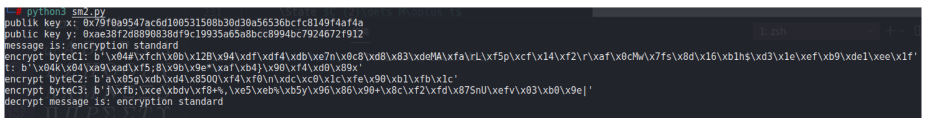
Input: 基点 G，阶数 n，密文 $C = C_1||C_3||C_2$，私钥 d_k

Output: 明文 M

1: $C_1, C_2, C_3 \leftarrow C$
2: $(x_1, y_1) \leftarrow bytestrToPoint(C_1)$
3: **if** not $(x_1, y_1) \in ECC$ **then**
4:     **return** $Wrong$
5: **end if**
6: $S \leftarrow [h]C_1$
7: **if** S is O **then**
8:     **return** $Wrong$
9: **end if**
10: $(x_2, y_2) \leftarrow [d_k]C_1$
11: $t \leftarrow KDF(x_2||y_2, klen)$
12: **if** t is 0 **then**

return $Wrong$
end if
$M' \leftarrow C_2 \oplus t$
$u \leftarrow Hash(x_2||M'|y_2)$
if $u \neq C_3$ **then**
return $Wrong$
end if
return $M \leftarrow M'$

3.4.4 测试样例及结果截图



3.4.5 总结

编程难点 为了 python 版本的一致性，本次实验首次转用 python3 bytes 类型处理数据而非转为比特字符串。因此编程实现并不直观

其他 sm2 的解密输出包含 3 个部分，其中 C1 与 C2 类似 Elgamal 密码体制，C3 为 Hash 值用于验证解密结果，这加大了密文篡改的困难性

4 总结

通过本次实验，我掌握椭圆曲线上的运算和常见的椭圆曲线密码算法，了解了椭圆曲线倍点运算的一些常用加速方法如双基链，半点运算。同时，了解基于 ECC 的伪随机数生成算法和基于椭圆曲线的商用密码算法 sm2。

5 思考题

1. 同等密钥长度下，RSA 算法比 ECC 算法更快，且实现简单
2. 与大数分解问题以及有限域上离散对数问题相比，椭圆曲线离散对数问题的求解难度要大得多。
3. 在同等安全需求下，ECC 比 RSA 需要的密钥规模要小得多。

4. 私钥产生简单。RSA 私钥产生时需要用到两个随机产生的大素数，除了需要保证随机性外，还需要用到素数判定算法，产生过程复杂且速度较慢。而 ECC 只需要生成随机数即可
5. 同等安全强度下，ECC 签名速度远胜于 RSA