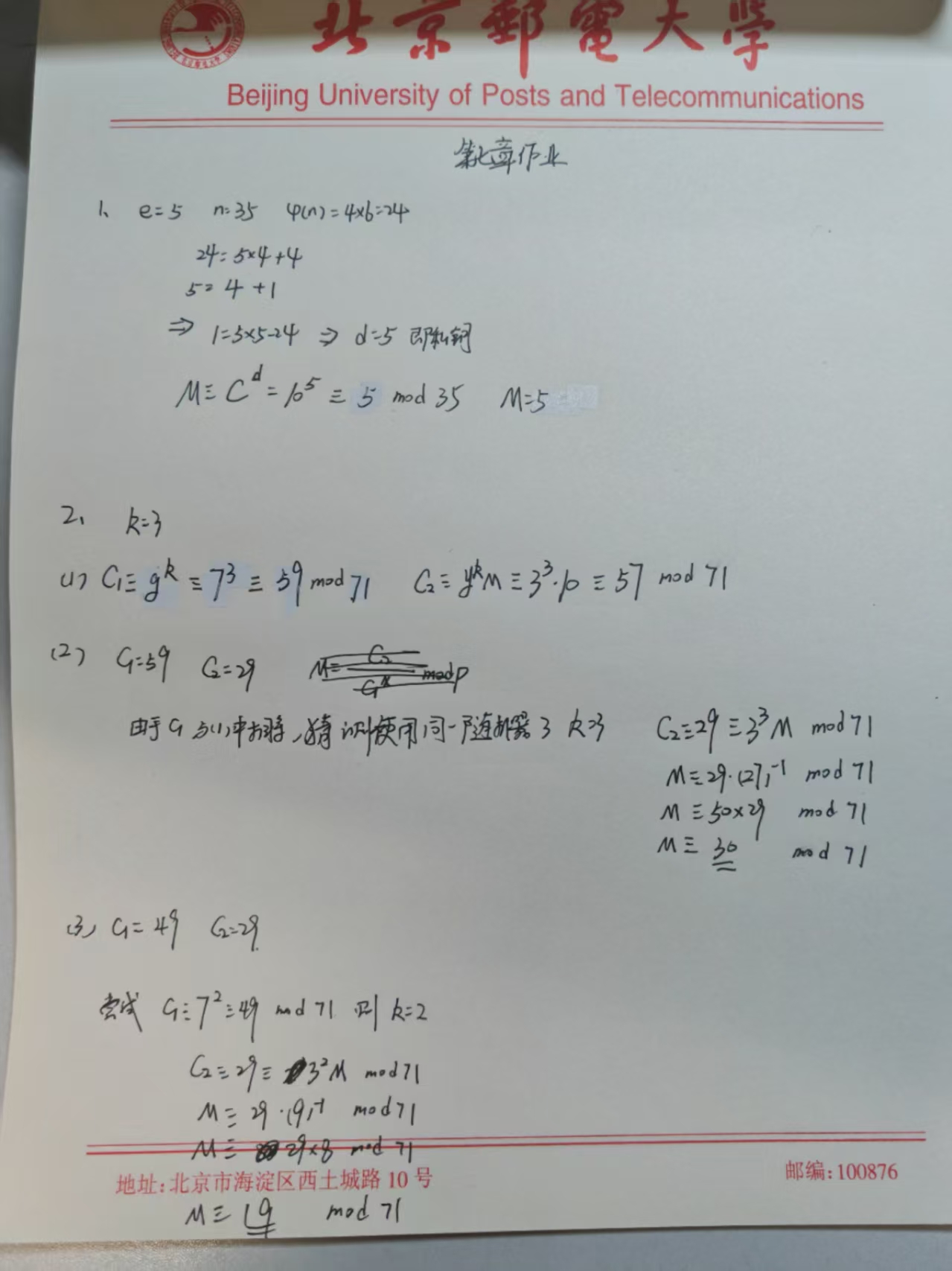
、

**3. 调研并实践生成大素数的方法**

生成大素数通常分为两个阶段：

（1）候选数生成（生成大整数）

使用随机函数生成一个固定长度的奇数（因为偶数除了 2 以外都不是素数）。

（2）素性测试（判断是否为素数）

主要使用概率性算法：

①Miller-Rabin素性测试

②Fermat 素性测试

③AKS 素性测试

在实际场景中主要使用Miller-Rabin算法，所以我们以此算法为例，进行大素数生成，以下为具体代码实现。

1. import random
2. *# Miller-Rabin 素性测试*
3. def is\_prime(n, k=40):
4. if n in (2, 3):
5. return True
6. if n <= 1 or n % 2 == 0:
7. return False
8. *# 将 n-1 写成 d \* 2^s*
9. s, d = 0, n - 1
10. while d % 2 == 0:
11. s += 1
12. d //= 2
13. for \_ in range(k):
14. a = random.randrange(2, n - 1)
15. x = pow(a, d, n)
16. if x == 1 or x == n - 1:
17. continue
18. for \_ in range(s - 1):
19. x = pow(x, 2, n)
20. if x == n - 1:
21. break
22. else:
23. return False
24. return True
25. *# 生成指定位数的大素数*
26. def generate\_large\_prime(bits=512):
27. assert bits >= 2
28. while True:
29. *# 生成一个奇数*
30. candidate = random.getrandbits(bits) | 1 | (1 << bits - 1)
31. if is\_prime(candidate):
32. return candidate
33. *# 示例：生成一个 512 位素数*
34. prime = generate\_large\_prime(512)
35. print("生成的大素数是：", prime)

**4.调研SM2加密标准的密钥生成、加密、解密详细步骤**

（1）SM2域参数

在进行SM2的密钥生成、加密、解密之前，需要首先确定一组椭圆曲线域参数。这些参数是算法的基础，通信双方必须使用相同的参数集。常用的SM2域参数是国家密码管理局公布的256位标准曲线参数，包括：

1.素数域 p： 定义有限域GF(p)。

2.椭圆曲线方程参数 a, b： 定义了Weierstrass形式的椭圆曲线方程 y² = x³ + ax + b (mod p)。

3.基点 G： 椭圆曲线上一个选定的点，是生成公私钥对的基础。G != 无穷远点O。

4.子群的阶 n： 基点 G 生成的循环子群的阶。n 是一个大素数。n \* G = O。

5.协因子 h： 椭圆曲线群的阶除以子群的阶，即 h = #E(F\_p) / n。对于SM2标准曲线，h=1。

这些参数通常是预先设定好的，使用者直接引用即可。

（2）密钥生成

1. 选择一条椭圆曲线和基点G（推荐使用国密推荐曲线参数）。

2. 随机生成一个私钥d（1 <= d <= n-1），n是椭圆曲线的阶。

3. 计算公钥P = d \* G。

输出：私钥d，公钥P。

（3）加密过程

设明文为M，加密过程如下：

1. 将明文M编码为点M'。

2. 生成随机数k（1 <= k <= n-1）。

3. 计算椭圆曲线点C1 = k \* G。

4. 计算共享密钥点 S = k \* P，设S = (x2, y2)。

5. 用KDF函数从x2和y2导出密钥t，t的长度等于明文长度。

6. 计算密文部分C2 = M XOR t。

7. 计算C3 = Hash(x2 || M || y2)。

8. 输出密文C = C1 || C2 || C3。

（4）解密过程

设密文为C = C1 || C2 || C3，解密过程如下：

1. 从密文中提取C1，C2，C3。

2. 验证C1是否在椭圆曲线E上。

3. 计算共享密钥点 S = d \* C1，设S = (x2, y2)。

4. 用KDF函数从x2和y2导出密钥t。

5. 计算明文M = C2 XOR t。

6. 验证Hash(x2 || M || y2) 是否等于C3，若相等则解密成功。

**5.调研密码库函数中模幂运算的快速实现方法**

（1）模幂运算简介

模幂运算是指计算a^b mod n，在密码学中广泛应用于RSA、DiffieHellman等算法。由于指数b可能非常大，直接计算效率低，因此需要高效算法实现。

（2）常用快速模幂算法

1.平方乘法

将指数b转为二进制，从高位到低位依次处理：

初始化结果为1。

遍历每一位：每次先平方结果，如果当前位为1则再乘以a。

每次运算后对n取模。

适用于大整数模幂的基本算法，简单高效。

2.Montgomery模幂算法

使用Montgomery乘法替代普通乘法，实现更高效的模乘。

无需除法操作，适用于硬件实现及大整数库。

通常结合平方乘法使用。

3.SlidingWindow算法

将指数b按窗口预处理，加快乘法操作。

空间换时间：预计算a的若干幂次以加快运算速度。

与平方乘法相比提高了效率，适用于固定基底或频繁模幂运算场景。

（3）密码库中实现示例

1.OpenSSL

使用BN\_mod\_exp函数实现模幂，内部使用Montgomery和窗口算法。

支持硬件加速和多种平台优化。

2.GMP（GNU MP Bignum Library）

提供mpz\_powm函数，支持平方乘法及优化路径选择。

适用于大整数密集运算。

3.Crypto++

提供ModularExponentiation接口，支持Montgomery和其它优化算法。

提供安全时间常数实现，防止侧信道攻击。

**6.调研椭圆曲线上点乘运算kG的快速实现方法**

（1）点乘运算简介

在椭圆曲线密码学（ECC）中，点乘是指对椭圆曲线基点G进行k次加法，记作kG。这是ECC的核心运算，广泛用于密钥生成、加密和签名中。由于k可能是大整数，必须使用高效算法加快计算。

（2）常用的快速点乘算法

1.传统二进制法（Binary Method）

将k转换为二进制，从高位到低位处理。

每一位对应一次“点加”或“点加+点倍”。

简单高效，是基本实现方式。

2.预计算窗口法（Windowed Method）

固定窗口法：将k分为固定大小的窗口，预先计算若干倍点以减少运行时点加次数。

滑动窗口法：动态选取窗口，结合预计算，提高效率，适用于重复基点情况。

3.Montgomery Ladder

实现常时间点乘，抗侧信道攻击。

每一步都执行一对点加和点倍操作。

安全性和效率兼顾。

4.双基点方法（Double-base Chains）

同时使用2和3为基进行分解，适合于特定硬件优化。

减少加法数量。

5.wNAF（Window Non Adjacent Form）

将k编码为稀疏的窗口非相邻表示，减少点加操作。

与窗口法结合使用效率更高。

（3）密码库中实现示例

1.OpenSSL

使用EC\_POINT\_mul实现点乘，内部支持多种算法：BN\_window\_bits决定窗口大小。

2.libsecp256k1

使用wNAF、预计算表等优化技术，针对secp256k1曲线高度优化。

3.Crypto++

提供EC2N和ECP类支持椭圆曲线点运算。