|  |
| --- |
| **Github（或者Coding）账号：** |
| **个人博客关于密码学实验的链接：** |
| **实验题目（中文）：**   1. **计算使得RSA不动点数量最小的 e的和；** 2. **实现RSA加密算法。** |
| **实验摘要（中文）：**  **本次实验主要探讨了RSA加密算法的关键步骤和实现细节。首先，我们探讨了如何选择合适的公钥e以最小化RSA不动点。我们发现，对于给定的素数p和q，存在多个满足条件的e值，这些e值可以使得RSA不动点数量达到最小。我们计算了所有这些e值的和，以进一步理解公钥选择对RSA加密效果的影响。**  **其次，我们深入研究了RSA加密的基本原理，包括素数的生成，模运算的应用，以及公钥和私钥的计算。我们发现，虽然RSA加密的数学原理相对简单，但在实现过程中需要处理一些复杂的问题，如大数运算和模逆运算。** |
| **题目描述（清楚描述题目中文，写出自己的理解，请勿复制原题目）**   1. **在RSA加密中，对于给定的素数p=1009和q=3643，找出所有满足1<e<φ(1009,3643)且gcd(e,φ)=1的e值，使得RSA不动点的数目最小，并求出这些e的和。RSA不动点是指满足m^e mod n=m的m值** 2. **使用编程实现RSA加密算法：在RSA加密中，首先生成两个随机素数p和q，然后计算n=p\*q和欧拉函数φ=(p-1)\*(q-1)。选择e=3，然后计算d=invmod(e, φ)。公钥为[e, n]，私钥为[d, n]。加密过程为c = m^e mod n，解密过程为m = c^d mod n。使用大素数进行测试，并将字符串转换为十六进制数进行加密。** |
| **过程（包括背景，原理：必要的公式，图表；步骤，如有必要画出流程图，给出主要实现步骤代码）**  第一题： 寻找满足特定条件的 e；  **第一步：首先需要对条件进行化简，即**    **其中。**  **因为p，q互质，则有**    **可得**    **或**    **由[3]知1.4式的解数分别为**    **因此满足1.2式的解数分别为**    **因此满足1.1式的解数为**    **第二步：**  **由式1.7知，给定e,p,q，就可计算出相应的RSA不动点的数目。因此设计算法步骤如下：**   1. **枚举找出所有与φ(n)互素的e。** 2. **枚举所有满足条件的e，计算RSA不动点的数目。** 3. **以RSA不动点的数目为键，累加变量为值，将每次的结果添加到字典中。** 4. **最后输出最小值的键对应的累加值。**   **代码以及运行结果如下，结果为399788195976：**    第二题：  **按部就班实现即可，其中求逆元用拓展欧几里得定理。**  **代码和运行结果如下：** |
| **总结（完成心得与其它，主要自己碰到的问题和解决问题的方法）**  **实验心得**  **本次实验让我对RSA加密算法有了更深入的理解。我学习了如何选择合适的公钥e以最小化RSA不动点，这是一个非常有趣的问题。我发现，对于给定的素数p和q，存在多个满足条件的e值，这些e值可以使得RSA不动点数量达到最小。这个发现让我对公钥选择的重要性有了更深的认识。**  **此外，我还深入研究了RSA加密的基本原理，包括素数的生成，模运算的应用，以及公钥和私钥的计算。我发现，虽然RSA加密的数学原理相对简单，但在实现过程中需要处理一些复杂的问题，如大数运算和模逆运算。这些问题的解决需要一定的数学知识和编程技巧。**  **在实验过程中，我还尝试了使用编程实现RSA加密算法。我学习了如何生成随机素数，如何计算欧拉函数，以及如何进行模运算。我还尝试了将字符串转换为十六进制数进行加密，这是一个非常实用的技巧。**  **总的来说，这次实验让我对RSA加密有了更深的理解。**  **主要问题**    **朴素算法效率相当低，后来查阅相关资料之后才发现有更好的解法。**  **另外一开始想着用拓展欧几里得求逆元，非常麻烦，做完了才发现在python3.8以上可以直接用pow函数求逆元。** |
| **参考文献（包括参考的书籍，论文，URL等，很重要）**   1. **[My C++ solution for Project Euler 182: RSA encryption](<https://euler.stephan-brumme.com/182/>)** 2. **[number theory - RSA fixed point - Mathematics Stack Exchange](<https://math.stackexchange.com/questions/1298664/rsa-fixed-point>)** 3. **[congruences - Number of solutions to $x^n \equiv 1 \mod p$ - Mathematics Stack Exchange](<https://math.stackexchange.com/questions/2549305/number-of-solutions-to-xn-equiv-1-mod-p>)** 4. **[algorithm - Modular multiplicative inverse function in Python - Stack Overflow](**[**https://stackoverflow.com/questions/4798654/modular-multiplicative-inverse-function-in-python#comment42039096\_9758173**](https://stackoverflow.com/questions/4798654/modular-multiplicative-inverse-function-in-python#comment42039096_9758173)**)** |