哈尔滨工业大学（深圳）

《密码学基础》实验报告

RSA 密码算法实验

学　院:　　计算机科学与技术

姓　名:　　李子韬

学　号:　　220110609

专　业:　　计算机科学与技术

日　期:　　2024-10-16

一、 实验步骤

请说明你的字符分组方式，以及关键的算法例如扩展欧几里德，素数检测，快速幂等函数的实现。

字符分组方式：两个字符的 ascii 码拼接为一个 6 位的十进制数字作为一个分组

- 具体来说，对于加密，先从明文取两个字符，利用 ord 取 ascii 码，再通过 ord0 * 1000 + ord1 实现拼接：

```
m_group = ord(pair[0]) * 1000 + ord(pair[1])
```

对于孤立的单个字符，使用\0 补全：

```
m_group = ord(plaintext[i]) * 1000
```

- 对于解密，也有类似操作，不过使用 chr 将 ascii 转为字符

```
decrypted_pair = mod_pow(num, d, n)
dechar_1 = chr(decrypted_pair // 1000)
dechar_2 = chr(decrypted_pair % 1000)
```

关键算法：

- 欧几里得算法求 gcd：根据辗转相除法的定义，最终 b==0 时 a 就是最后的 gcd，也就是最后一步前的 b 必能整除 a(最极端情况下 b=1)

```
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a
```

- 素数检测：使用 Miller-Rabin 算法检测素数，思路就是先做初始检查（小边界和提高效率），之后分解 n-1=2^r * d 的形式，进行 k=10 轮平方测试，每一轮测试中，依次检验平方模是否为 n - 1，如果没有(for 循环正常结束)，则调用 for-else 语句的 else 返回合数的判断，否则算暂时通过，进行下一轮的测试；

```python
def is_prime(n, k=10):  # 使用Miller-Rabin算法检测素数
    if n <= 1:
        return False
    if n <= 3:
        return True
    if n % 2 == 0:
        return False

    # 写成 n - 1 = d * 2^r
    r, d = 0, n - 1
    while d % 2 == 0:
        d //= 2
        r += 1

    # k 次测试
    for _ in range(k):
        a = random.randint(2, n - 2)
        x = pow(a, d, n)
        if x == 1 or x == n - 1:
            continue

        for _ in range(r - 1):
            x = pow(x, 2, n)
            if x == n - 1:
                break
        else:
            return False
    return True
```

- **扩展欧几里德算法求逆：思 n 路是每一轮的辗转相除需要得到 q 整除结果，然后利用 x(n)=x(n-2)-qx(n-1)迭代即可，不过需要注意初始设置 x 的值即可（此处假定 a 是更大的那个数，故 x0 = 0， x1 = 1，其中 x0 比 x1 更晚更新）**

```python
def mod_inv(a, m):
    m0, x0, x1 = m, 0, 1
    if m == 1:
        return 0
    while a > 1:
        q = a // m
        m, a = a % m, m
```

```
        x0, x1 = x1 - q * x0, x0
    if x1 < 0:
        x1 += m0
    return x1
```

- **快速幂函数：从末位开始解析，每解析一个 bit 就让 b 平方模一次，同时让 exp 右移 1 位，实现快速幂计算**

```
def mod_pow(b, exp, mod):
    res = 1
    b = b % mod  # b < mod

    while exp > 0:
        if (exp % 2) == 1:
            res = (res * b) % mod

        exp = exp // 2
        b = (b * b) % mod

    return res
```

算法总体框架：

1. 生成 p、q，计算 n=pq、phi(n)=(p-1) * (q-1)
2. 生成公钥 e，mod_inv 计算模 phi(n)逆元私钥 d
3. 对每个明文分组，加密
4. 解密，解分组
5. 输出结果

以下是实现：生成密钥

```python
def generate_keys():
    p = gen_prime(64)
    q = gen_prime(64)
    n = p * q
    phi_n = (p - 1) * (q - 1)
    e = gen_prime(64)
    d = mod_inv(e, phi_n)

    return {
        'p': p,
        'q': q,
        'n': n,
        'e': e,
        'd': d,
        'phi(n)': phi_n
    }
```

分组加密和解密:

```python
def encrypt(plaintext, n, e):
    ciphertext = []
    start_time = time.time()

    for i in range(0, len(plaintext), 2):
        if i + 1 < len(plaintext):
            pair = plaintext[i:i + 2]
            m_group = ord(pair[0]) * 1000 + ord(pair[1])
            ciphertext.append(mod_pow(m_group, e, n))
        else:
            m_group = ord(plaintext[i]) * 1000
            ciphertext.append(mod_pow(m_group, e, n))

    end_time = time.time()
    encryption_time = end_time - start_time

    return ciphertext, encryption_time

def decrypt(ciphertext, n, d):
    decrypted = ''
    start_time = time.time()

    for num in ciphertext:
        decrypted_pair = mod_pow(num, d, n)
        dechar_1 = chr(decrypted_pair // 1000)
        dechar_2 = chr(decrypted_pair % 1000)
        decrypted += dechar_1 + dechar_2

    end_time = time.time()
    decryption_time = end_time - start_time

    return decrypted, decryption_time
```

结果输出部分:

```python
    # 从文件读取明文
    with open('lab2-Plaintext.txt', 'r') as file:
        plaintext = file.read()

    # 生成密钥
    keys = generate_keys()

    # 将公钥和私钥写入文件
    with open('public_key.txt', 'w') as pub_file:
        pub_file.write(f"n: {keys['n']}\n")
        pub_file.write(f"e: {keys['e']}\n")

    with open('private_key.txt', 'w') as priv_file:
        priv_file.write(f"d: {keys['d']}\n")

    # 执行RSA加密
    ciphertext, encryption_time = encrypt(plaintext, keys['n'], keys['e'])

    # 将密文写入文件
    with open('ciphertext.txt', 'w') as file:
        for num in ciphertext:
            file.write(f"{num}\n")  # 每个密文占一行

    # 执行RSA解密
    decrypted_text, decryption_time = decrypt(ciphertext, keys['n'], keys['d'])
```

```python
    # 输出结果
    print("p:", keys['p'])
    print("q:", keys['q'])
    print("Public key written to 'public_key.txt'")
    print("Private key written to 'private_key.txt'")
    print("Ciphertext written to 'ciphertext.txt'")
    print("Decrypted text:", decrypted_text)
    print("Encryption time (seconds):", encryption_time)
    print("Decryption time (seconds):", decryption_time)
```

二、实验结果与分析

程序正确运行的结果截图，包括p, q, n,e, d,$\phi(n)$这些参数的值，可输出加密解密时间作为时长参考。

```
[Ionic from Terence_Lee][0.076s][RAM: 9/15GB][Thursday at 11:20:30 PM]
[E:\Code\Github-Projects\HITsz-Cryptography-Project\code\Lab2]
└─Δ python .\rsa.py
p: 15505713786394599769
q: 2159946762929740493
Public key written to 'public_key.txt'
Private key written to 'private_key.txt'
Ciphertext written to 'ciphertext.txt'
Decrypted text: 2002 A.M. TURING AWARD. RSA, an acronym for Rivest, Shamir and Adleman, uses algorithmic number theory to provide an efficient realization
 of a public-key cryptosystem, a concept first envisioned theoretically by Whitfield Diffie, Martin Hellman and Ralph Merkle. RSA is now the most widely u
sed encryption method, with applications throughout the Internet to secure on-line transactions. It has also inspired breakthrough work in both theoretica
l computer science and mathematics.
Encryption time (seconds): 0.009480714797973633
Decryption time (seconds): 0.014608383178710938
[Ionic from Terence_Lee][0.083s][RAM: 9/15GB][Thursday at 11:20:44 PM]
```

```
[ Ionic from  Terence_Lee][ 0.083s][ RAM: 9/15GB][ Thursday at 11:20:44 PM]
└[E:\Code\Github-Projects\HITsz-Cryptography-Project\code\Lab2]
└─Δ python .\rsa.py
p: 16715162189744618681
q: 17485513048613895809
Public key written to 'public_key.txt'
Private key written to 'private_key.txt'
Ciphertext written to 'ciphertext.txt'
Decrypted text: 2002 A.M. TURING AWARD. RSA, an acronym for Rivest, Shamir and Adleman, uses algorithmic number theory to provide an efficient realization
 of a public-key cryptosystem, a concept first envisioned theoretically by Whitfield Diffie, Martin Hellman and Ralph Merkle. RSA is now the most widely u
sed encryption method, with applications throughout the Internet to secure on-line transactions. It has also inspired breakthrough work in both theoretica
l computer science and mathematics.
Encryption time (seconds): 0.005001544952392578
Decryption time (seconds): 0.01032710075378418
[ Ionic from  Terence_Lee][ 0.058s][ RAM: 9/15GB][ Thursday at 11:26:30 PM]
```

可见，两次实验的 p，q 值均在 64bit 数字的范围，加密的结果是 Ciphertext，解密结果与原文相同，算法运行正常；加密解密的的时间都极短(解密 0.01s 左右)

其中，压缩包内附有密文，可查看

三、总结

1、实验过程中遇到的问题有哪些？你是怎么解决的？

1）一开始扩展欧几里得算法的迭代写反了，后续通过单元测试调初始值和数的顺序解决；

2）k 较低(一开始设置的是 3)存在解密崩掉成乱码的可能性，其原因可能是 p, q 不是素数但是经过了 is_prime 的考验，后续增加 k 的轮次再也没出现过；

2、关于本实验的意见或建议。

建议加入线上测试和验收平台，就像 MIT/Stanford 实验那样