实验三 RSA

课程名称:《信息安全》SOFT130018.01

任课老师:李景涛

助教:雷哲

实验目的

- 实现 RSA 加密
- 理解分解模数破解 RSA 的过程
- 理解公共模数攻击

实验内容

RSA

RSA (Rivest–Shamir–Adleman) is one of the first public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and distinct from the decryption key which is kept secret (private).

The idea of RSA is based on the fact that it is difficult to factorize a large integer. The public key consists of two numbers where one number is multiplication of two large prime numbers. And private key is also derived from the same two prime numbers. So if somebody can factorize the large number, the private key is compromised.

Key Generation

1. Pick two large prime numbers p and q at random, multiply them together to produce the modulus n.
2. Calculate the Euler totient function of n: $\phi(n) = (p-1)(q-1)$.
3. Choose an integer *encryption exponent* e such that $1 < e < \phi(n)$ and $\gcd\big(e, \phi(n)\big) = 1$; that is, e and $\phi(n)$ are coprime.
4. Determine *decryption exponent* d as $d \equiv e^{-1}(mod\ \phi(n))$; that is, d is the modular multiplicative inverse of e modulo $\phi(n)$.
5. Return public key (n, e) and private key (n, d) .

Encryption & Decryption

First turns message M (strictly speaking, the un-padded plaintext) into an integer m (strictly speaking, the padded plaintext), such that $0 \leq m \leq n$ by using an agreed-upon reversible protocol known as a *padding scheme*. (实现时不用考虑 padding，如实现可有适当加分。)

Compute the ciphertext c, using public key e :

$$c \equiv m^e \ (mod\ n)$$

Recover m from c by using private key exponent d by computing:

$$m \equiv c^d \ (mod\ n)$$

Breaking RSA

There are many attacks against RSA. The following lists two simple one which related to the experiment.

Factoring the Public Key

If the modulus n can be factored into p and q, then d can be recovered through the same process as in *key generation* to decrypt the ciphertext.

Common Modulus Attack

If the attacker knows:

$$ct_1 = E_{(n,e_1)}(M)$$

$$ct_2 = E_{(n,e_2)}(M)$$

He can recover the plaintext if $\gcd(e_1, e_2) = 1$ and $\gcd(ct_2, n) = 1$

> ***Bezout's Theorem**: if there are integers a and b, which are not both zero, then there are integers x and y such that:*
>
> $xa + yb = gcd\ (a, b)$

If $\gcd(e_1, e_2) = 1$, then it has integers x and y such that:

$$xe_1 + ye_2 = 1$$

By using the **Extended Euclidean algorithm** one can find and x and y.

Then its easy to show that the plaintext can be recovered as follows:

$C_1^x * C_2^y = (M^{e_1})^x * (M^{e_2})^y$

$= M^{e_1 x} * M^{e_2 y}$

$= M^{e_1 x + e_2 y}$

$= M^1$

$= M$

all math performed in the common modulo n.

Normally y will be a negative integer. As such, C^y must be evaluated as follows:

Let $y = -a$

$C_2^y = C_2^{-a}$

$= (C_2^{-1})^a$

$= (\frac{1}{C_2})^{-y}$

$\gcd(ct_2, n) = 1$，so C_2 is invertible in mod n.

Thus, in order to practically recover the plaintext message the attacker has to calculate:

$C_1^x * (C_2^{-1})^{-y}$

实验步骤

1.实现 RSA 算法

实现如下几个函数：

```
def multiplicative_inverse(e, phi):
    '''
    extended Euclid's algorithm for finding the multiplicative
```

```
inverse
    '''
    # WRITE YOUR CODE HERE!
def key_generation(p, q):
    # WRITE YOUR CODE HERE!
```

```
def encrypt(pk, plaintext):
    # WRITE YOUR CODE HERE!
```

```
def decrypt(sk, ciphertext):
    # WRITE YOUR CODE HERE!
```

2.分解模数破解 RSA

有两个文件 secret.enc pubkey.pem

1. 使openssl解析pubkey.pem中参数，得到e, n.

```
openssl rsa -pubin -text -modulus -in warmup -in
pubkey.pem
```

2. 将n由十六进制转换为十进制
3. 对大整数n进行分解，得到p and q(http://www.factordb.com/index.php/或者调用库函数/自己编程实现)
4. 求出d

```
import gmpy2
phi_n= (p - 1) * (q - 1)
d = gmpy2.invert(e, phi_n)
```

或者调用上述实验中自己实现的求模逆元函数

5. 读出secret.enc->bytes2num->用私钥d解密密文->nums2str->输出明文

```
def bytes2num(b):
    s = '0x'
    for x in b:
        tmp = str(hex(x))[2:]
        if len(tmp)==2:
            pass
        else:
```

```
            tmp = '0' +tmp
        s += tmp
    num = int(s, 16)
    return num

def num2str(n):
    tmp = str(hex(n))[2:]
    if len(tmp)%2 == 0:
        pass
    else:
        tmp = '0' +tmp
    s = ''
    for i in range(0, len(tmp), 2):
        temp = tmp[i] + tmp[i+1]
        s += chr(int(temp, 16))
    return s
```

或者调用库函数。

```
fi = open('secret.enc', 'rb')
cipher = fi.read()
cipher = bytes2num(cipher)
fi.close()
```

> *注：此处 secret.enc 为 PKCS#1 v1.5 padding 填充模式加密的结果，要求输入的明文块必须比 RSA 钥模长(modulus) 短至少 11 个字节，即 RSA_size(rsa) – 11*

3.公共模数攻击

在 common_modulus.py 中实现攻击脚本，并破解如下参数，输出明文：

```
n =
1031090659023346202226101162008793963504256027939117020091876799039690801944735604259018655534860183205031069083254290258577291605287053538752280231959857465853228851714786887294961873006234153079187216285516823832102424110934062954272346111907571393964363630079343598511602013316604641904852018969178919051627
ct1 =
27986802045732033022419852993635525978880040058860117071655484864259480070279380433837578120046031676442023798716710398905392478828569924500223842772568159517424084570259026599281616052048845009994936090641131136657248157049330825538797172631844962895927650351182341902533
```

```
288671645286092267149867544688152842
e1 = 15
ct2 =
35879228223724294438913809689907680896276707028526527015080082428780
80366814212667585371488412653137377940402491828233679160578665939001
65895629806685459677372266012188946509296546002164648649470846122279
78977961749829638118214589326703093221940569232753322324189970925416
535971639377507300741236055601665808
e2 = 13
```

实验要求和评分

- 编程语言、编译运行时，所用工具等实验环境原则上不限。但需要完成同等任务，并在实验报告中写明。
- 评分内容如下：

内容	总分 100
multiplicative_inverse	10
key_generation	10
encrypt&decrypt	10
factoring attack	30
common modulus attack	30
Document(实验报告)	10

实现 padding 会有适当加分

实验提交

- ➢ 独立完成 project，不分组
- ➢ 提交内容清单：

1. 实验报告（按**实验报告模板**书写，提交 pdf 格式文件，命名格式：学号+姓名+实验三.pdf）
2. 项目源代码（命名格式：学号_rsa.py，学号_common_modulus.py）
3. 鼓励录制短视频，介绍代码结构、演示运行结果、分析计算量等

- ➢ 提交方式：所有提交内容以压缩文件形式上传 elearning 提交（命名格式：学号+姓名+实验三）
- ➢ 提交截止时间：5 月 3 日 23:59 前

参考资料：

- https://ctf101.org/cryptography/what-is-rsa/
- RSA 的原理与实现
- RSA加密解密原理深度剖析（附CTF中RSA题型实战分析）
- RsaCtfTool

- http://www.factordb.com/index.php
  https://gmpy2.readthedocs.io/en/latest/
- ssh-keygen - Generate a New SSH Key
- Twenty Years of Attacks on the RSA Cryptosystem by Dan Boneh