实验三 RSA

课程名称：《密码学基础》 COMP130069.01
任课老师： 李景涛
助教： 潘天雨

You can also access this document here:
http://xpty.xyz/post/rsa/

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 $\gcd(e, \phi(n)) = 1$; that is, e and $\phi(n)$ are coprime.
4. Determine *decryption exponent* d as $d \equiv e^{-1} \pmod{\phi(n)}$; 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 $0 \leq m < n$ 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:

$$c \equiv m^e \pmod{n}$$

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

$$m \equiv c^d \pmod{n}$$

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 x and y.

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

$$\begin{aligned} C_1^x * C_2^y &= (M^{e_1})^x * (M^{e_2})^y \\ &= M^{e_1 x} * M^{e_2 y} \\ &= M^{e_1 x + e_2 y} \\ &= M^1 \\ &= M \end{aligned}$$

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 $y = -a$

$$\begin{aligned} C_2^y &= C_2^{-a} \\ &= \left(C_2^{-1}\right)^a \\ &= \left(\frac{1}{C_2}\right)^{-y} \end{aligned}$$

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

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

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

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. 使用 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 和 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. 读入`secret.enc` → bytes2num →用私钥d解密密文→ num2str→ 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()
```

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 =
103109065902334620226101162008793963504256027939117020091876799039690801944735604259018655534860183205031069083254290258577291605287053538752280231959857465853228851714786887294961873006234153079187216285516823832102424110934062954272346111907571393964363630079343598511602013316604641904852018969178919051627
ct1 =
891652677575920569984305152733932368246756182096281206369764153245416087776393129063287618961963355639189958130654460971115112996225835595212794161183442626595878734974943260495650772083917978408463647115805683221822096164874540866208321669127946228263575291651851223106585538881756296044865442655939860652568
e1 = 15
ct2 =
201352435731023123654894455551794717110502373577715319445651972079090568567914526600715339813518136154060426868286668836027435656926804662251288789077763396384988888366429201228092846902432706751150972015124343370541335540941476313114064244336027572156250928469136937900784244586243692285757760167435329469 34
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：（100分是本实验的总分，本学期五个实验各100分，学期末会加权得出实验分总分并汇入总成绩）

内容	总分100
multiplicative_inverse	10
key_generation	10
encrypt&decrypt	10
factoring attack	30

内容	总分100
common modulus attack	30
Document(实验报告)	10

> *实现padding 会有适当加分*

实验提交

- 分组规则:不分组,即独立完成project
- 完成提交截止时间: **5月10日23:00前**
- 提交内容清单:需要提交
    - 实验报告 （按照**实验报告模板**书写，要求提交`pdf`格式文件）（命名格式：`姓名+学号+实验三.pdf`）
    - 项目源代码 (命名格式：`学号_rsa.py`，`学号_common_modulus.py`)
- 提交方式:
  所有提交清单中的文件,压缩后, eLearning提交
- 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

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