RSA加/解密器

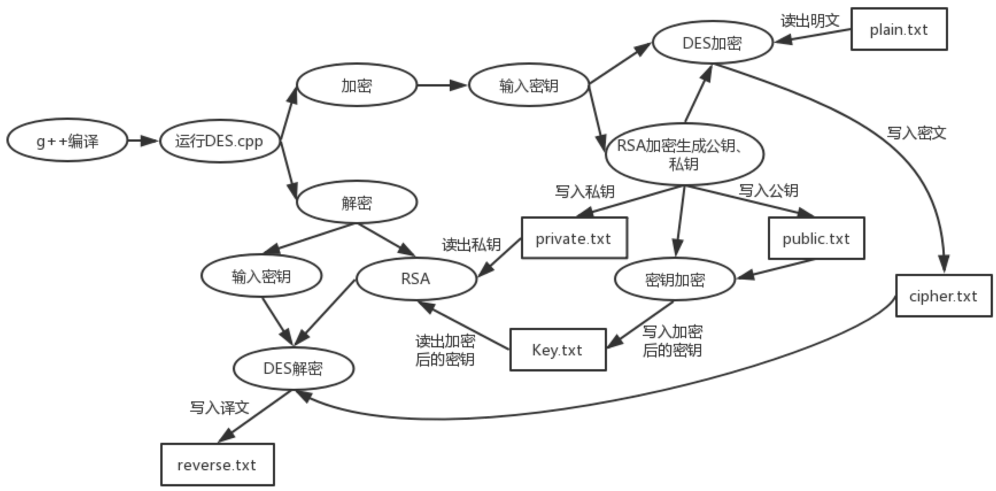
罗翔 17307130191

RSA加/解密器

操作流程

```
PS G:\Scholarly Resource\Computer science\信息安全原理\RSA> g++ -o DES .\DES.cpp .\RSA.cpp
PS G:\Scholarly Resource\Computer science\信息安全原理\RSA> .\DES.exe
Welcome to the DES-cryption system!
Input: 0 indicates encryption, while 1 means decryption.
0
Input the keys.
0100100011111100000000001111110000000000111111000000000011111100
Input: 0 indicates directly encrypting with keys while 1 means working with public keys through RSA.
1
PS G:\Scholarly Resource\Computer science\信息安全原理\RSA> .\DES.exe
Welcome to the DES-cryption system!
Input: 0 indicates encryption, while 1 means decryption.
1
Input 0 indicates directly decrypting with keys while 1 means working with private keys through RSA.
1
PS G:\Scholarly Resource\Computer science\信息安全原理\RSA>
```

Windows系统下，进入文件所在目录后，可以在命令行中通过g++进行编译，然后如图运行DES.exe文件，首先选择加密还是解密，如果选择加密则首先需要输入密钥，然后选择直接用DES加密或者使用RSA生成公钥和私钥再进行加密。加密时会从当前目录下的 "plain.txt" 文件中读出明文，经加密后将密文写入 ”cipher.txt“ 文件中。如果调用RSA则会在"public.txt"中生成公钥以及大质数的乘积，在"private.txt"中生成私钥，大质数的乘积以及两个大质数，同时经公钥加密后的密钥会写入” Key.txt“中。解密时，如果选择直接解密则需要再次输入密钥，如果选择RSA则会从 ” Key.txt“，"private.txt"中分别读出经公钥加密后的密钥以及私钥，将密钥解密后用来将”cipher.txt"中的加密内容解密并写入"reverse.txt"中。流程图如下：



基本原理

随机生成大素数p，q，（根据素数定理有一个64位的整数位质数的概率$1/ln(2^{64})\approx 1/44$）并求$n=pq$以及欧拉函数$\varphi(n)=(p-1)(q-1)$，然后随机选取整数d使得$(d,\varphi(n))=1$，然后求出e使得$de\equiv 1 \mod \varphi(n)$，将e（公钥）和n公布，d（私钥）自己保存。加密明文m时有

$$c=m^e \mod n$$

解密密文c时有

$$m=c^d \mod n$$

下对解密运算的正确性证明：

对任意$m<n$

- 若$(m,n)=1$，由欧拉定理有

  $$m^{\varphi(n)}\equiv 1 \mod n$$

  又由$de\equiv 1 \mod \varphi(n)$有，$de=k\varphi(n)+1$，所以

  $$\begin{aligned} c^d \mod n &= (m^e)^d \mod n \\ &= m^{de} \mod n \\ &= m^{k\varphi(n)+1} \mod n \\ &= (m^{\varphi(n)})^k m \mod n \\ &= m \mod n \end{aligned}$$

- 若$(m,n)>1$，又$n=pq$，所以(m,n)必含有p或q中一个，设为p，可有$\exists h$使得$m=h\cdot p$，且$(m,q)=1$，由欧拉公式有

$$m^{\varphi(q)}\equiv 1 \mod q$$

又因为q为素数，可有$\varphi(q)=q-1$

$$\begin{aligned} m^{k\varphi(n)} \mod q &= m^{k(p-1)(q-1)} \mod q \\ &= (m^{\varphi(q)})^{k(p-1)} \mod q \\ &= 1 \mod q \end{aligned}$$

所以有

$$
\begin{aligned}
m^{k\varphi(n)+1} \mod n &= (w\cdot q+1)m \mod n \\
&= (wqhp+m) \mod n = m \mod n
\end{aligned}
$$

加密

随机数检验

加密前需要先随机生成大素数p，q。可以通过Miller-Rabin检验法检验是否为素数，考虑：

若p为素数，则可有

$$p-1=2^k q, 2\nmid q$$

设a与p互质，则下述条件中必有一条成立：

- $a^q \equiv 1 \mod p$
- a^q，a^{2q}，…，$a^{2^{k-1}q} \equiv -1 \mod p$ 中有一等式成立
  因为$a^{p-1}=a^{2^k q} \equiv 1 \mod p$，所以$a^{2^{k-1}q} \equiv \pm 1 \mod p$，即上述所有项均为1或出现-1。

快速幂运算

考虑$a^b=a^{b_0+2b_1+\cdots+2^{r-1}b_{r-1}}=a^{b_0}(a^2)^{b_1}\cdots(a^{2^{r-1}})^{b_{r-1}}$

$$(a^{2^i})^{b_i}=\begin{cases}1 & b_i=0\\ a^{2^i} & b_i=1\end{cases}$$

因此最多需要2r-2次模乘法运算，当设定公钥为65537时，则可以有效减少乘法次数（$32\to 18$次）。

Euclidean算法

由上所说，因为预先设定了公钥，所以可以通过Euclidean算法求出e在模$\varphi(n)$域下的逆元d（私钥）。
如果$gcd(a,b)=gcd(b,a\%b)=1$，即a，b互质，可有$ax+by=1$。又$gcd(a,b)=1$，所以有

$$
\begin{aligned}
ax_1+by_1=gcd(a,b)=1\\
bx_2+(a\%b)y_2=gcd(b,a\%b)=1
\end{aligned}
$$

所以有

$$
\begin{aligned}
ax_1+by_1 &= bx_2+(a-(a/b)*b)y_2\\
&= ay_2+b(x_2-(a/b)y_2)
\end{aligned}
$$

即$x_1=y_2$， $y_1=x_2-(a/b)y_2$。因此可以构造两个数列$t_0,t_1,\cdots,t_m$和$s_0,s_1,\cdots,s_m$使得
$r_j=s_jr_0+t_jr_1$，$r_{m-2}=q_{m-1}r_{m-1}+r_m$，$r_0=a,r_1=b$

$$
\begin{aligned}
t_0=0,t_1=1,t_j=t_{j-2}-q_{j-1}t_{j-1}(j>1)\\
s_0=1,s_1=0,s_j=s_{j-2}-q_{j-1}s_{j-1}(j>1)
\end{aligned}
$$

其中$q_j=\lfloor r_{j-1}/r_k\rfloor$当$r_j\neq 0$

解密

解密时，因为e（公钥）定在65537较小，所以一般情况下，d（私钥）会比较大，做幂运算的速度仍然较慢，所以可以考虑利用中国剩余定理来提高解密速度，具体操作如下。

设密文为$c\equiv m^e \mod n, n=pq$

设$c_1\equiv c \mod p, c_2\equiv c \mod q$

$d_1 \equiv d \mod (p-1), d_2 \equiv d \mod (q-1)$

$m_1 \equiv c_1^{d_1} \mod p, m_2 \equiv c_2^{d_2} \mod q$

可由中国剩余定理求解同余方程组$m \equiv m_1 \mod p, m \equiv m_2 \mod q$

下证其正确性

$$\begin{aligned} m &= c^d \mod n \\ &= c^d \mod p \\ &\equiv c_1^d \mod p \\ &= c_1^{k(p-1)+d_1} \mod p \\ &= c_1^{d_1} \mod p \equiv m_1 \end{aligned}$$