DoubleS

发现其实加密方式就是用一个32阶方阵去乘以一个列向量并得到一个列向量
$$\LARGE
\begin{bmatrix}
1&name_1&name_1^2&\cdots&name_1^{31}\
1&name{2}&name{2}^2&\cdots&name{2}^{31}\
\vdots&\vdots&\vdots&\ddots&\vdots\
1&name{32}&name{32}^2&\cdots&name{32}^{31}\
\end{bmatrix}
\cdot
\begin{bmatrix}
s_1\s_2\\vdots\s{32}
\end{bmatrix}
=\begin{bmatrix}
r_1\r_2\\vdots\r{32}
\end{bmatrix}

可以直接用线性代数的方法得到$$ $[r_1 \quad r_2 \quad \cdots \quad r_{32}]^T sage$代码

```
def power(a,b):
    ret=1
    for i in range(b):
        ret*=a
    return ret
from Crypto.Util.number import *
path="outputs"
f=open(path,'rb')
temp=f.read().split(b'\n')
f.close()
name=[]
r=[]
for i in temp[:-1]:
    name.append(bytes_to_long(i.split()[0]))
    r.append(i.split()[1].decode())

A=matrix(ZZ,32,32)
B=matrix(ZZ,32,1)
for i in range(32):
    for j in range(32):
        A[i,j]=power(name[i],j)
    B[i,0]=r[i]
S=A.inverse()*B
for i in range(32):
    try:
        print(long_to_bytes(int(S[i,0])).decode(),end='')
    except:
        break
```

DoubleSS

这似乎是个非预期解$\cdots$
$$\LARGE
\begin{bmatrix}
1&name_1&name_1^2&\cdots&name_1^{31}\
1&name{2}&name{2}^2&\cdots&name{2}^{31}\
\vdots&\vdots&\vdots&\ddots&\vdots\
1&name{31}&name{31}^2&\cdots&name{31}^{31}\
\end{bmatrix}
\cdot
\begin{bmatrix}
s_1\s_2\\vdots\s{32}
\end{bmatrix}
=\begin{bmatrix}
r_1\r_2\\vdots\r{31}
\end{bmatrix}

比$DoubleS$少了一行，如果我们能把矩阵补为32×32,就可以解得$[r_1 \quad r_2 \quad \cdots \quad r_{32}]^T$ 如果我们能令：$$

$$\begin{bmatrix}1 & name_1 & nam\\ 1 & name_2 & nam\\ \vdots & \vdots & \vdots\\ 1 & name_{31} & nam\\ 1 & 0 & 0\end{bmatrix}$$

即可求解
因为$\Large r_1$相对 $\Large name_1^{31}$很小，范围为587202560~603979776\
在范围中随便取几个值，可求得
$\large
\begin{bmatrix}
r_1&r_2&\cdots&r{32}
\end{bmatrix}^T
$
的猜测值。其中$\large r_i$为实数。
最后，我们可以发现，随机猜测的$\large r_1$，对于求得$\large r_2$~$\large r{31}$影响不大。
因此，我们可以求得$r_2$为bytes_to_long(b'S_c0')
sage代码

```
def power(a,b):
    ret=1
    for i in range(b):
        ret*=a
    return ret
import random
from Crypto.Util.number import *
path="C:\\Users\\94974\\Desktop\\DoubleS\\outputs"
f=open(path,'rb')
temp=f.read().split(b'\n')
f.close()
name=[]
r=[]
for i in temp[:-1]:
    name.append(bytes_to_long(i.split()[0]))
    r.append(i.split()[1].decode())

A=matrix(ZZ,32,32)
B=matrix(ZZ,32,1)
for i in range(31):
    for j in range(32):
        A[i,j]=power(name[i],j)
    B[i,0]=r[i]
A[31,1]=1
B[31,0]=bytes_to_long(b'S_c0')
S=A.inverse()*B
for i in range(32):
    try:
        print(long_to_bytes(int(S[i,0])).decode(),end='')
    except:
        print(int(S[i,0]))
```

Copiano

低指数加密的RSA，暴力破解即可
python代码

```
from Crypto.Util.number import *
import gmpy2 as g
def dec(x,N):
    while 1:
        if g.iroot(x,3)[1]:
            return int(g.iroot(x,3)[0])
        else:
            print(g.iroot(x,3)[1])
            x+=N
N=
e=3
c=
x_list=
m=[]
cnt=0
for i in range(8):
    cipher=bytes_to_long(c[i*256:(i+1)*256])
    m.append(dec(cipher,N))
    print(long_to_bytes(m[-1]^x_list[i]).decode(),end='')
```