实验原理

一．RC4算法简介

RC4是一种专有的字节流加密算法，设计于1987年。随着算法渐渐受到公众的注意，RSA实验室把该算法说明当作商业机密。实现者应该与RSA实验室就此事进行商议。但是在1994年，该算法被反推导出来并匿名公开。

二．RC4算法加密过程 动画演示

由于RC4算法好实现，它也好描述。RC4的基本思想是生成一个叫密钥流的伪随机序列字节流，然后与数据相异或（XOR）。异或运算是一种按位运算，其常用数学符号⊕来表示，其规则如下：

1⊕1=0　1⊕0=1　0⊕1=1　0⊕0=0。

异或运算有如下性质：如a⊕b=c则有c⊕a=b　c⊕b=a　即a⊕b⊕b=a。

RC4正是利用上面的运算性质实现了数据的加密解密：

加密：明文⊕随机数=密文

解密：密文⊕随机数=明文

“随机”是指在攻击者看来是随机的，而连接的两端都能够产生相同的“随机”值处理每一个字节。因此它被称为伪随机，是由RC4算法生成的。

伪随机密钥流最重要的性质是，只要知道用于生成字节流的密钥，你就可能算出序列中的下一个字节。如果你不知道密钥，它看起来就真的是像随机的，注意，异或操作完全隐藏了明文值。即使明文是一长串的0，在攻击者看来密文依然是随机数。

RC4有两个阶段：密钥安装阶段和伪随机生成阶段。

密钥安装阶段是用密钥安装算法建立一个由0～255排列组成的字节阵列，就是说，所有的数字都出现在阵列中但顺序已被打乱，阵列中的排列叫做S-Box，最初有0～255顺序排列而成，然后S-Box通过下列过程进行重排。首先，第二个256字节的阵列，或者叫做K-Box，用密钥填充，按照需求被重复填充满整个阵列。现在让每一个S-Box中的字节与S-Box中的另一个字节进行互换。从第一个字节开始，作如下操作：

j = j +（S-Box中的第一个字节的值）+（K-Box第一个字节的值），其中j是一个单字节的值忽略任何加法计算中的溢出。

现在把j作为S-Box的索引，把该位置上的值与第一个位置上的值进行互换，重复这样的操作255次，直到S-Box中的每一个字节都被换过。算法的程序伪代码描述如下。

```
i=j=0;
for(i=0; i<256; ++i)
{
    j=(j+S[i]+K[i])mod 256;
    Swap(S[i], S[j]);
}
```

伪随机生成阶段是在S-Box初始化后，是S-Box中更多字节的交换，每次迭代（R）生成一个伪随机字节。迭代操作程序伪代码如下。

```
i=j=k=u=0;
i=(i+1) mod 256;
j=(j+S[i]) mod 256;
Swap(S[i], S[j]);
K=(S[i]+S[j]) mod 256;
R=S[k];
```

明文第u个字节与R异或生成密文。每字节的明文与RC4算法产生的R值相异或，生成密文。注意完成整个过程只需要字节长度的加法和互换，这对于计算机是非常简单的操作。

理论上讲，RC4不是一个完全安全的加密系统，因为它生成的是伪随机密钥流，不是真正的随机字节，但如果在协议中正确的使用，对于我们的应用来说，它的确是足够安全的。