DES算法设计

15331159 李沁航

算法原理概述

DES属于块加密算法，通过将输入的明文按照64位为块长，分成多块，然后每块依次加密解密。 DES的密钥为56位，从而保证了其难以被破解，即使已经将内部工作原理完全公开。 算法的原理如下：

设置密钥

首先选择一个64位的密钥，并将它的每第8位放弃，用于奇偶检测，总共放弃8位之后剩余的56位为使用的密钥。

初始置换

将一块64位的明文参照Initial Permutations表进行置换，即例如有 `publicText[i] = j;` 则将明文的第i位替换成之前第j位的内容。 然后将置换后产生的块分为两个部分，分别为LPT，以及RPT，每个各32位。

DES round加密

在此过程中共需要循环16次，每一次将56位的密码分成两个28位的，然后按照以下的表

轮次	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
位数	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

按照次序将两边共同向左移动1或者2位。然后再按照compression premutation的表进行选择，得到剩下的48位密钥。

拓展

在之前得到的左明文（LPT）和右明文（RPT）两个32位的块中，我们需要将RPT进行拓展。方法为：

- 将明文分为8块有4位的段
- 将4位拓展为6位
    - 具体参照右明文拓展置换表
- 然后将在round加密中得到的48位密钥与48位右明文进行XOR操作
- 之后进行S盒替换

S盒替换

将48位密钥和铭文XOR操作后的48位输入使用8个S盒，将每6位缩小至4位，然后得到32位的块。

- 操作方法为 6位的第1以及最后一位表示行数，第2-5位勇用于表示列数，然后选取S盒中该行中对应列的值输出，该值转化为4bit。最终组合起来得到32位的块。

P盒置换

依照表格，将前一步得到的32位置换，得到新的32位的块。

异或和交换

将左明文（LPT）与上一步得到的新块进行XOR运算，得到的结果位新的右明文（RPT），然后再将LPT替换为旧的RPT，于是得到结果。

最终置换

经过16轮之后，按照表（最终置换表）进行最终替换，然后得到加密后的密文。

在此算法中，加密的算法也同样适用于解密。解密时的密钥使用顺序与加密时相反。

总体结构

```
void bitsCopy(bool *dataOut,bool *dataIn,int num);
void bitToHex(char *dataOut,bool *dataIn,int num);   //依照bit转换为16进制数
void hexToBit(bool *dataOut,char *dataIn,int num);   //16进制数转换为bit
void bitToByte(char *dataOut,bool *dataIn,int num);  //将bit转换为byte
void keyInBit(bool *keyOut, char* keyIn, int num);   //将密钥转换为64位的bit
void tableP(bool* dataOut, bool* dataIn, const char* table, int num); //根据表来进行对数据置换的函数
void circulate(bool* dataIn, int len, int num); //在16次循环中将LPT盒RPT循环置换的函数
void Xor(bool* dataA, bool* dataB, int num); //异或所用到的函数

void sBox(bool dataOut[32], bool dataIn[32]); //S盒置换
void textExpand(bool dataIn[32], bool dataI[48]); //32-48位拓展函数
void setKey(char key[8]); //设置密钥
void desEncrypt(char dataOut[8], char dataIn[8]); //加密
void desDecrypt(char dataOut[8], char dataIn[8]); //解密
```

数据结构

使用char* 来储存明文以及key，使用bool数组来储存bit的值，具体的表格可以参见附件中的table.h。 bool数组自定义于bool.h头文件中。

类C语言算法过程

可以参见附件中的cpp文件。 其中最早实验时，是将明文加密后依旧按照byte的类型输出出来，导致密文由于编码的问题无法正确显示，所以更改为了按照16进制数输出，使得密文更容易读懂。

所有的置换所用到的表格，均来自于《密码与网络安全》第二版 Atul Kahate著的书中