# DES 实验报告

16326109 谢昆成

# 算法原理

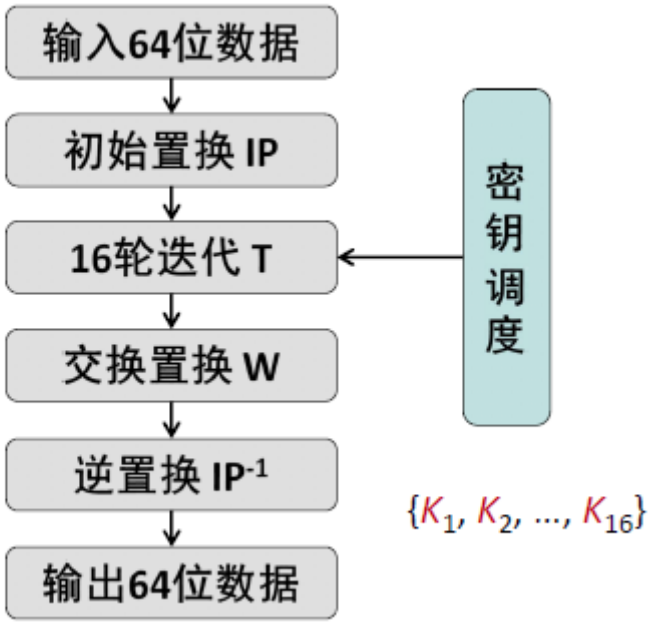
DES 是一种典型的块加密方法：它以64位为分组长度，64位一组的明文作为算法的输入，通过一系列复杂的操作，输出同样64位长度的密文。

DES 采用64位密钥，但由于每8位中的最后1位用于奇偶校验，实际有效密钥长度为56位。密钥可以是任意的56位的数，且可随时改变。其中极少量的数被认为是弱密钥，但能容易地避开它们。所有的保密性依赖于密钥。

DES 算法的基本过程是换位和置换。

# 总体结构

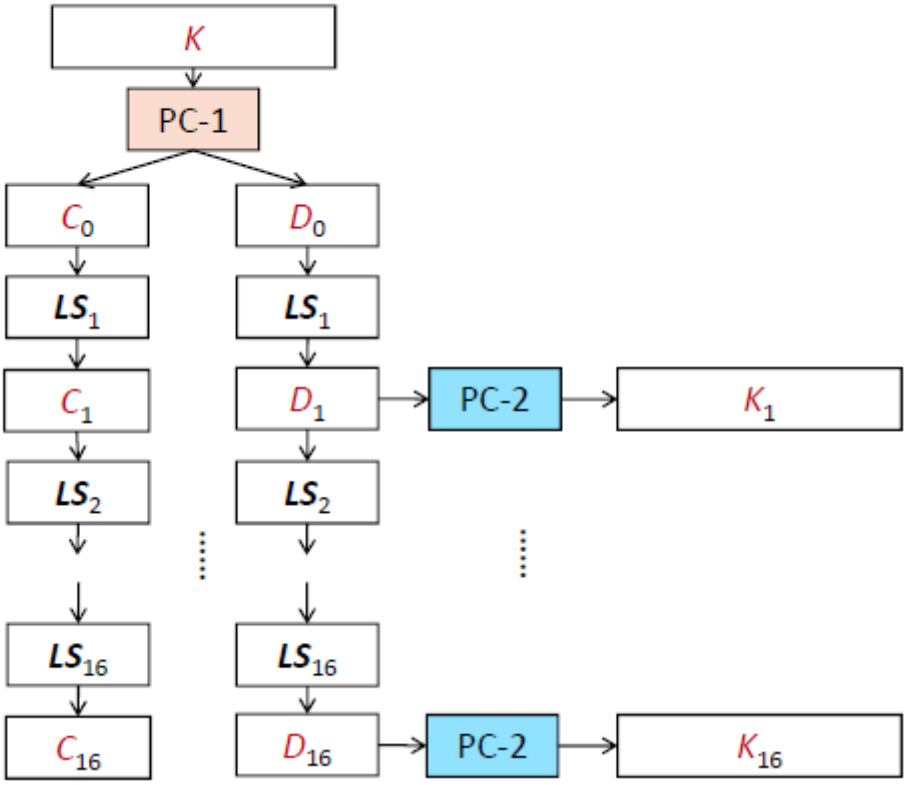
## DES的总体结构




- 加密过程 $C = E_k(M) = IP^{-1} \cdot W \cdot T_{16} \cdot T_{15} \cdots T_1 \cdot IP(M)$. M 为算法输入的64位明文块； $E_k$ 描述以K 为密钥的加密函数，由连续的过程复合构成； IP 为64位初始置换； $T_1, T_2, \ldots, T_{16}$ 是一系列的迭代变换； W 为64位置换，将输入的高32位和低32位交换后输出； $IP^{-1}$ 是IP 的逆置换； C 为算法输出的64位密文块。
- 解密过程$M = D_k(C) = IP^{-1} \cdot W \cdot T_1 \cdot T_2 \cdots T_{16} \cdot IP(C)$.

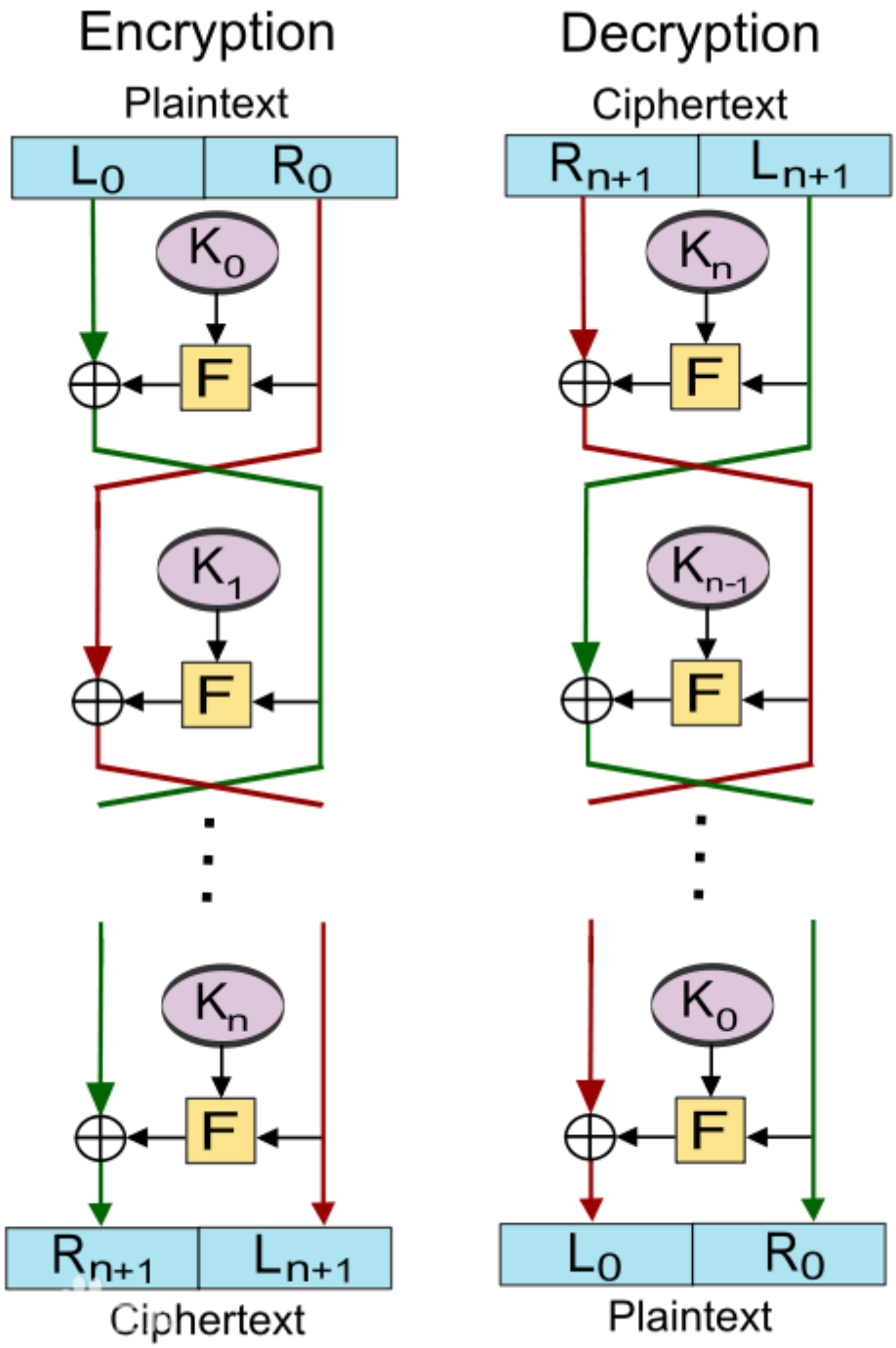
加密过程和解密过程算法一致，只是子密钥的调度顺序相反。

## 子密钥的生成算法




1. 对K 的56个非校验位实行置换PC-1，得到$C_0D_0$，其中$C_0, D_0$分别由PC-1 置换后的前28位和后28位组成。
2. 计算$C_i = LS_i(C_i - 1)$和$D_i = LS_i(D_i - 1)$
   - 当i =1, 2, 9, 16 时，$LS_i(A)$ 表示将二进制串A 循环左移一个位置；否则循环左移两个位置。
3. 对56位的$C_iD_i$ 实行PC-2 压缩置换，得到48位的$K_i$, i = i+1。
4. 如果已经得到$K_{16}$，密钥调度过程结束；否则转(2)。

## Feistel轮函数




1. 将长度为32位的串$R_{i-1}$ 作E-扩展，成为48位的串$E(R_{i-1})$；
2. 将$E(R_{i-1})$ 和长度为48位的子密钥$K_i$ 作48位二进制串按位异或运算，$K_i$ 由密钥K生成；
3. 将(2) 得到的结果平均分成8个分组，每个分组长度6位。各个分组分别经过8个不同的S-盒进行6-4 转换，得到8个长度分别为4位的分组；
4. 将(3) 得到的分组结果顺序连接得到长度为32位的串；
5. 将(4) 的结果经过P-置换，得到的结果作为轮函数$f(R_{i-1}, K_i)$ 的最终32位输出。

# 模块分解

据DES的主体结构，其可以分解成三个大的模块：整体算法框架、子密钥生成和Feistel轮函数。

整体算法框架负责整个DES算法，包括密钥的调度。

子密钥生成将生成所需的16个48位密钥。

Feistel轮函数由输入产生Feistel函数的结果。

# 设计思路

我们可以对密钥定义一个数据结构以保存密钥的c、d和k。

用unsigned char 代表一个字节。

本人采用一种创新的方式：将轮换、字节拆解（8转6，4转8）和移位按照位映射的方式统一处理。如此，用映射数组代替操作代码，大大减少了代码量。

加密和解密调用同一函数`process`，只是通过传入不同参数而使调用子密钥的顺序相反。

通过模块分解，将DES分解为几大函数：`bitMapping`, `generateSubKey`, `feistel`和`process`，得以较好的完成算法流程。

# 运行结果

将`test.txt`中的文件用程序加密后解密得到`plain.txt`，对比发现二者一致，说明DES算法编写正确。

# C语言源代码

源程序在src目录下，主要包含DES.c、DES.h和main.c代码文件。`des.key`存放密钥。

在编写程序过程中，遇到了不少坑。分享如下：

- malloc申请内存空间时没有初始化内容，用calloc则可
- strncpy遇到字符`'\0'`会截断拷贝，造成拷贝内容过短，改用memcpy
- char在向右移位时会补符号位，造成错误。改用unsigned char代表一个字节。

# 编译运行方法

编译方式

```
gcc ./DES.c ./main.c -std=c99 -o a
```

运行格式

```
./a inputFilename keyFilename outputFilename mode
```

mode为1表示加密，mode为0表示解密

例：

将要加密的内容存入`test.txt`，将密钥存放于`des.key`。编译运行即在`cipher.txt`中得到密文，在`plain.txt`中得到解密后的明文。在`log.txt`存放运行的中间结果。

加密

```
./a test.txt des.key cipher.txt 1 >log.txt
```

解密

```
./a cipher.txt des.key plain.txt 0 >log.txt
```