DES 加密算法的过程分析研究

◆余启航　李斌勇　杨雄凯　姚　瑶

（成都信息工程大学网络空间安全学院　四川　610225）

摘要：本文围绕 DES 算法的加密问题，分析了 DES 算法的明文分组加密和子密钥生成过程。针对 DES 所涉及的核心算法模块，分别对 IP 初始置换、子密钥获取、E 盒与 S 盒扩展、异或运算、P 盒置换和逆初始置换模块进行了深入地研究与设计。在此基础上给出 DES 的部分核心算法的输入与输出实现，并对其安全性进行了分析。本文所开展的研究，为深入理解 DES 加密算法和常规应用加密问题，提供了可行的参考。

关键词：DES；子密钥；置换；加密算法

0　引言

DES 加密解密算法最初由美国 IBM 公司研究人员所设计发明，且为第一个公开的商用密码算法标准，自诞生以来便得到了 ISO 的一致认可。DES 是分组密码算法的典型代表，它的明文分组长度为 64bits，密钥长度为 64bits，其中包括有 8bits 的奇偶校验，因此有效密钥长度为 56bits。DES 加密解密算法使用的过程相同，且可以随时均都可以进行变动。它们其中有极少数被认为是易破解的弱密钥，但是很容易抛开它们不使用，因此其自身安全性主要依赖于有效密钥。由于 DES 算法使用最大为 64bits 的逻辑运算以及标准算术，它的子密钥产生较为容易，可以适用于当前大部分计算机当中，因此近三十多年以来，其在保密通信密码算法的研究使用中，扮演着极其重要的作用。

1　DES 算法分析

DES 算法加密过程首先先对明文分组进行操作，需要加密的明文分为每块 64bits 的固定大小。如图 1 所示左右两部分分别为 64bits 的明文分组加密过程和其 16 个子密钥生成的过程。

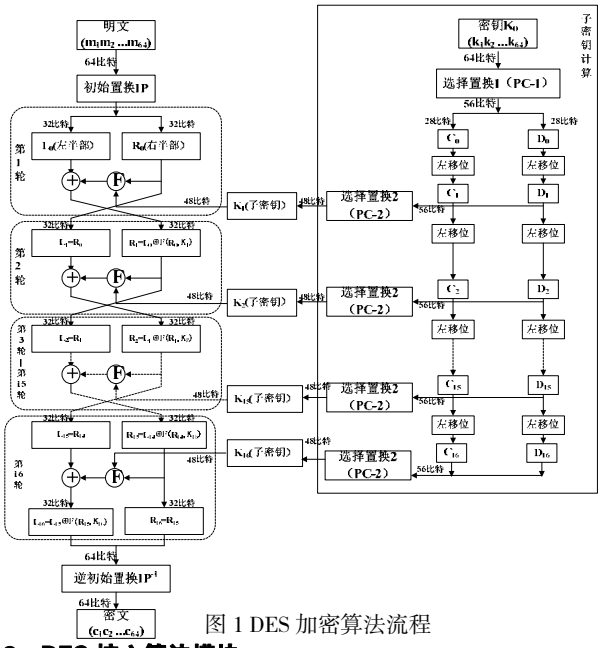

图 1 DES 加密算法流程

2　DES 核心算法模块

2.1 IP 初始置换

IP 初始置换，在第一轮运算之前执行，对输入的分组采用如表 1 所示的 IP 初始变换，按照从左向右、从上向下进行置换。

表 1 IP 初始置换

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

2.2 子密钥获取流程

子密钥的获取流程如图 2 所示。

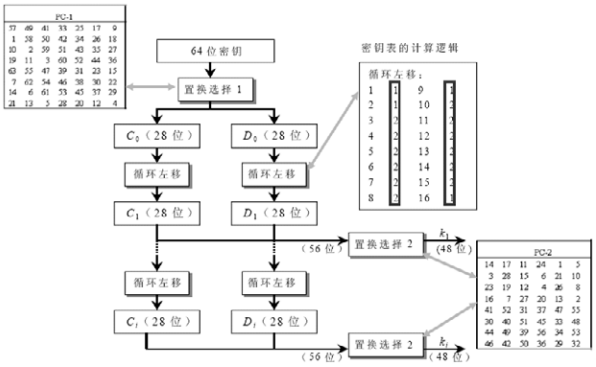

图 2 子密钥获取流程

Step1：此处，用户输入 64 位的密钥。根据密钥置换表 PC–1，将 64 位变成 56 位密钥（此处去掉了奇偶校验位）。

Step2：PC–1 置换得到的 56 位密钥。此处密钥被分为前 28 位 C0 和后 28 位 D0。分别对它们进行循环左移，C0 左移得到 C1，D0 左移得到 D1。

Step3：将 C1 和 D1 合并变成 56 位。然后通过 PC–2 表进行压缩置换，得到此轮的 48 位子密钥 K1。

Step4：再对 C1 和 D1 进行相同的左移和压缩置换，获取下一轮的子密钥……一共进行 16 轮，于是可以得到 16 个 48 bits 的子密钥。

2.3 E 盒扩展

E 盒扩展置换，则是将右半部分 32bits 按照 8 行 4 列方式依次排列，得到一个 8*4 的二维矩阵，然后根据如表 2 所示的 E 盒扩展置换表扩展为 8*6 的二维矩阵。

表 2　E 盒扩展

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

2.4 异或运算

将 P 盒置换的结果与最初的 64bits 分组的左半部分异或，然后左、右半部分交换，接着开始另一轮。

2.5 S 盒扩展

当产生了 48bits 密钥后就可以和明文进行异或运算，便可得到 48bits 的密文。再开始下轮的 S 盒迭代运算，其功能是把 6bit 数据变为 4bits 数据，每个 S 盒是一个 4 行、16 列的表。每个 S 盒的使用方法为：S 盒收到 6bits 的输入，6bits 的第 1 个 bit 和最后 1 个 bits 构成的 2 位二进制为该 S 盒行号，中间的 4bits 二进制为该 S 盒的列号，然后根据行号和列号查 S 盒定义表得到对应的值（通常为十进制），该值就是 S 盒变换的输出，并转化为二进制。

2.6 P 盒置换

S 盒代替运算之后，输出 32bits，作为 F 函数最后一个变换 P 盒置换的输入。将该 32bits 位数据进行 P 盒置换，置换后得到一个仍然是 32 bits 的结果，此处可得 F 函数的输出值。

2.7 逆初始置换

DES 完成 16 轮变换后，得到 64bits 数据作为 IP-1 逆初始置换的输入，经过 IP-1 逆初始置换表（如表 3 所示），64bits 输入数据位置重新编排，就得到 64bits 的密文。

表 3 逆初始置换

40	8	48	16	56	24	64	32	39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30	37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28	35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26	33	1	41	9	49	17	57	25

3 DES 核心算法实现

3.1 通过 UnitoHex 函数将 unicode 字符转换为 16 进制

```
def UnitoHex(string):
return_string=''
for i in string:
return_string+="%02x"%ord(i)
return return_string
```

输入：unicode 字符串

输出：十六进制数据流

3.2 通过_CodeIP 函数对密文或明文初始置换

```
def _CodeIP(self,code):
changed_code=''
for i in range(64):
changed_code+=code[ip[i]-1]
return changed_code
```

输入：明文或密文

输出：对明文或密文初始置换后的 64bit 数据

3.3 通过_KeyIP 函数对密钥初始置换。

```
def _KeyIP (self,key):
changed_key=''
for i in range(56):
changed_key+=key[pc1[i]-1]
return changed_key
```

输入：初始密钥

输出：对密钥初始置换后的 56bit 数据

3.4 通过_EBox 函数进行 E 盒扩展置换

```
def _EBox(self,code):
return_list=''
for i in range(48):
return_list+=code[e[i]-1]
return return_list
```

输入：32bit 数据

输出：E 盒扩展置换后的 48bit 数据

3.5 通过_Sbox 函数进行 S 盒代替选择置换

```
def _SBox(self, key):
return_list=''
for i in range(8):
row=int(str(key[i*6])+str(key[i*6+5]),2)
raw=int(str( key[i*6+1])+str(key[i*6+2])+str(key[i*6+3])+str(key[i*6+4]),2)
return_list+=self._toByte(s[i][row][raw],4)
return return_list
```

输入：48bit 数据

输出：S 盒代替选择置换后的 32bit 数据

3.6 通过_Pbox 函数进行 P 盒置换

```
def _PBox(self,code):
return_list=''
for i in range(32):
return_list+=code[p[i]-1]
return return_list
```

输入：32bit 数据

输出：P 盒置换后的 32bit 数据

3.7 通过_Xor 函数对数据和子密钥进行异或

```
def _Xor(self,code,key):
code_len=len(key)
return_list=''
for i in range(code_len):
if code[i]==key[i]:
return_list+='0'
else:
return_list+='1'
return return_list
```

输入：E 盒扩展置换后的 48bit 数据和 PC-2 的 48bit 子密钥。

输出：异或后的 48bit 数据。

4 安全性分析

由于 DES 算法中只用到 64bits 密钥中的其中 56bits 密钥，而第 8、16、...64bits 中的 8 个 bits 并未参与 DES 运算，可以发现即 DES 的安全性是基于除 8、16、...64bits 以外的其余 56bits 的排列组合才可以得到保证的。因此，在实际进行保密通信中，应尽量避免使用第 8、16、24 等 bits 作为有效数据位进行加密解密，以免在进行保密通信的系统中产生数据被破译的隐患。

5 结束语

DES加密解密算法较为复杂，其中用到多次不同置换方式，以及进制的转换和有效位的采用等，这给实现此算法带来了一定的难度。但是也正是因为此，它的安全性相比其他一般密码算法的安全性较高。在未来进一步研究中，应着力研究DES有效位的使用，以及完善加密解密的过程和简便其算法的实现过程，以增强其安全性，使之能更加广泛地解决日常应用加密问题。

参考文献：

[1]刘浪，周新卫.基于 DES 对称加密体制的探讨[J].科技广场，2012.

[2]方亮.DES 加密算法 IP 模块实现[D].电子科技大学，2011.

[3]张芯苑.基于 python 的加密解密算法实现与研究[J].纺织报告，2017.

[4]徐洪波，李颖华.DES 加密算法在保护文件传输中数据安全的应用[J].信息网络安全，2009.

[5]张温泉，赵红敏，郝晓东.一种高速高安全性的 DES 算法设计[J].微电子学与计算机，2014.

[6]张峰，郑春来，耶晓东. DES 加密算法的 FPGA 实现[J].现代电子技术，2008.

基金项目：四川省教育厅重点项目(17ZA0069)、成都信息工程大学科研基金资助项目(KYTZ201618)。