



南开大学

Nankai University

南 开 大 学

网络空间安全学院

密码学实验报告二

Lab2 分组密码算法 DES

学号： 2013484

姓名： 张世伟

年级： 2020 级

专业： 信息安全-法学

2022 年 11 月 24 日

摘 要

本文 C++ 语言实现了 DES 算法，并检测计算了雪崩效应

关键词: C++，DES，雪崩效应

目 录

摘要	I
第 1 节 AES 加解密算法	1

第 1 节 AES 加解密算法



图 1.1: AES 算法流程图

```
Microsoft Visual Studio 调试器
第0组，加密成功
第0组，解密成功

第1组，加密成功
第1组，解密成功

改变明文
change plaintext[0]: 61
change plaintext[1]: 58
change plaintext[2]: 63
change plaintext[3]: 60
change plaintext[4]: 72
change plaintext[5]: 66
change plaintext[6]: 68
change plaintext[7]: 65
average: 64.125

改变密钥
change key[0]: 71
change key[1]: 65
change key[2]: 69
change key[3]: 60
change key[4]: 70
change key[5]: 62
change key[6]: 66
change key[7]: 71
average: 66.75

D:\Study\Programs\AES_CRYPT0\x64\Debug\AES_CRYPT0.exe (进程 20616)已退出，代码为 0。
```

图 1.2: 输出结果演示

$N_b=N_k=4, N_r=10$

2 组样例加解密全部正确，改变 plaintext1 位，8 次得到的雪崩效应平均值是 64.125，改变 key1 位，8 次计算得到的雪崩效应平均值是 66.75

接下来概括性写一下中间生成结果（以第 0 组数据为例）

```

Microsoft Visual Studio 调试器
key:
0x00 0x01 0x20 0x01 0x71 0x01 0x98 0xae 0xda 0x79 0x17 0x14 0x60 0x15 0x35 0x94
plaintext:
0x00 0x01 0x00 0x01 0x01 0xa1 0x98 0xaf 0xda 0x78 0x17 0x34 0x86 0x15 0x35 0x66
ciphertext:
0x6c 0xdd 0x59 0x6b 0x8f 0x56 0x42 0xcb 0xd2 0x3b 0x47 0x98 0x1a 0x65 0x42 0x2a

加密结果:
0x6c 0xdd 0x59 0x6b 0x8f 0x56 0x42 0xcb 0xd2 0x3b 0x47 0x98 0x1a 0x65 0x42 0x2a
第0组, 加密成功
解密结果:
0x00 0x01 0x00 0x01 0x01 0xa1 0x98 0xaf 0xda 0x78 0x17 0x34 0x86 0x15 0x35 0x66
第0组, 解密成功

key:
0x2b 0x7e 0x15 0x16 0x28 0xae 0xd2 0xa6 0xab 0xf7 0x15 0x88 0x09 0xcf 0x4f 0x3c
plaintext:
0x32 0x43 0xf6 0xa8 0x88 0x5a 0x30 0x8d 0x31 0x31 0x98 0xa2 0xe0 0x37 0x07 0x34
ciphertext:
0x39 0x25 0x84 0x1d 0x02 0xdc 0x09 0xfb 0xdc 0x11 0x85 0x97 0x19 0x6a 0x0b 0x32

加密结果:
0x39 0x25 0x84 0x1d 0x02 0xdc 0x09 0xfb 0xdc 0x11 0x85 0x97 0x19 0x6a 0x0b 0x32
第1组, 加密成功
解密结果:
0x32 0x43 0xf6 0xa8 0x88 0x5a 0x30 0x8d 0x31 0x31 0x98 0xa2 0xe0 0x37 0x07 0x34
第1组, 解密成功

改变明文average: 59.125

```

图 1.3: 样例加解密结果

```

1 S,rS,rC和测试样例见data.h
2 #include "data.h"
3 using namespace std;
4 string int2binstr(int text[4][4]) {
5     string result;
6     for (int i = 0; i < 4; i++) {
7         for (int j = 0; j < 4; j++) {
8             string str = "00000000";
9             int temp = text[j][i];
10            for (int k = 7; k >= 0; k--) {
11                str[k] = '0' + temp % 2;
12                temp /= 2;
13            }
14            result += str;
15        }
16    }

```

```
17         return result;
18     }
19     void binstr2int(int text[4][4], string str) {
20         unsigned char* output = new unsigned char[16];
21         for (int i = 0; i <= 15; i++) {
22             int start = i * 8;
23             int temp = 0;
24             for (int j = start; j <= start + 7; j++) {
25                 int each = 1;
26                 for (int s = 1; s <= 7 - j + start; s++) {
27                     each *= 2;
28                 }
29                 if (str[i] == '1') {
30                     temp += each;
31                 }
32             }
33             output[i] = temp;
34         }
35         for (int i = 0; i < 4; i++) {
36             for (int j = 0; j < 4; j++) {
37                 text[j][i] = output[j * 4 + i];
38             }
39         }
40     }
41     //基本运算
42     int mult(int a, int b)
43     {
44         int third = b & 0x8; //b>=8
45         int second = b & 0x4;
46         int first = b & 0x2;
47         int firstMod = b % 2;
48         int res = 0;
49         if (third)
50         {
51             int temp = a;
52             for (int i = 1; i <= 3; ++i)
```

```
53         {
54             temp = temp << 1;
55             if (temp >= 256)
56             {
57                 temp = temp ^ 0x11b;
58             }
59         }
60         temp = temp % 256;
61         res = res ^ temp;
62     }
63     if (second)
64     {
65         int temp = a;
66         for (int i = 1; i <= 2; ++i)
67         {
68             temp = temp << 1;
69             if (temp >= 256)
70             {
71                 temp = temp ^ 0x11b;
72             }
73         }
74         temp = temp % 256;
75         res = res ^ temp;
76     }
77     if (first)
78     {
79         int temp = a;
80         temp = temp << 1;
81         if (temp >= 256)
82         {
83             temp = temp ^ 0x11b;
84         }
85         temp = temp % 256;
86         res = res ^ temp;
87     }
88     if (firstMod)
```

```
89     {
90         res = res ^ a;
91     }
92     return res;
93 }
94 void KeyExpansion(int key[4][4], int Exp[11][4][4]) //密钥拓展
95 {
96     for (int i = 0; i < 4; ++i)
97     {
98         for (int j = 0; j < 4; j++)
99         {
100             Exp[0][i][j] = key[j][i];
101         }
102     }
103     for (int i = 1; i < 11; ++i)
104     {
105         for (int j = 0; j < 4; ++j)
106         {
107             int temp[4];
108             if (j == 0)
109             {
110                 temp[0] = Exp[i - 1][3][1];
111                 temp[1] = Exp[i - 1][3][2];
112                 temp[2] = Exp[i - 1][3][3];
113                 temp[3] = Exp[i - 1][3][0];
114                 for (int k = 0; k < 4; ++k)
115                 {
116                     int m = temp[k];
117                     int row = m / 16;
118                     int col = m % 16;
119                     temp[k] = S[row][col];
120                     if (k == 0)
121                     {
122                         temp[k] = temp[k] ^ rC[i]
123                     }
124                 }
125             }
126         }
127     }
128 }
```



```
125         }
126         else
127         {
128             temp[0] = Exp[i][j - 1][0];
129             temp[1] = Exp[i][j - 1][1];
130             temp[2] = Exp[i][j - 1][2];
131             temp[3] = Exp[i][j - 1][3];
132         }
133         for (int x = 0; x < 4; x++)
134         {
135             Exp[i][j][x] = Exp[i - 1][j][x] ^ temp[x];
136         }
137     }
138 }
139 }
140 void ByteSub(int input[4][4], int type) //字节变换
141 {
142     for (int i = 0; i < 4; i++)
143     {
144         for (int j = 0; j < 4; j++)
145         {
146             int temp = input[i][j];
147             int row = temp / 16;
148             int col = temp % 16;
149             if (type == 1)
150             {
151                 input[i][j] = S[row][col];
152             }
153             if (type == 0)
154             {
155                 input[i][j] = rS[row][col];
156             }
157         }
158     }
159 }
160 void ShiftRow(int input[4][4], int type) //行移位变换
```

```
161 {
162     for (int i = 0; i < 4; i++)
163     {
164         for (int j = 0; j < i; j++)
165         {
166             if (type == 1)
167             {
168                 int temp = input[i][0];
169                 input[i][0] = input[i][1];
170                 input[i][1] = input[i][2];
171                 input[i][2] = input[i][3];
172                 input[i][3] = temp;
173             }
174             else
175             {
176                 int temp = input[i][3];
177                 input[i][3] = input[i][2];
178                 input[i][2] = input[i][1];
179                 input[i][1] = input[i][0];
180                 input[i][0] = temp;
181             }
182         }
183     }
184 }
185 void MixColumn(int input[4][4], int type)//列混合变换
186 {
187     for (int i = 0; i < 4; i++)
188     {
189         int t0 = input[0][i];
190         int t1 = input[1][i];
191         int t2 = input[2][i];
192         int t3 = input[3][i];
193         if (type == 1)
194         {
195             input[0][i] = mult(t0, 2) ^ mult(t1, 3) ^ t2 ^ t3;
196             input[1][i] = t0 ^ mult(t1, 2) ^ mult(t2, 3) ^ t3;
```

```

197         input[2][i] = t0 ^ t1 ^ mult(t2, 2) ^ mult(t3, 3)
198         input[3][i] = mult(t0, 3) ^ t1 ^ t2 ^ mult(t3, 2)
199     }
200     else
201     {
202         input[0][i] = mult(t0, 14) ^ mult(t1, 11) ^ mult(t2, 9) ^ mult(t3, 13)
203         input[1][i] = mult(t0, 9) ^ mult(t1, 14) ^ mult(t2, 13) ^ mult(t3, 11)
204         input[2][i] = mult(t0, 13) ^ mult(t1, 9) ^ mult(t2, 11) ^ mult(t3, 14)
205         input[3][i] = mult(t0, 11) ^ mult(t1, 13) ^ mult(t2, 14) ^ mult(t3, 9)
206     }
207 }
208 }
209 void AddRoundKey(int input[4][4], int key[4][4]) // 密钥加
210 {
211     for (int i = 0; i < 4; ++i)
212     {
213         for (int j = 0; j < 4; j++)
214         {
215             input[i][j] = input[i][j] ^ key[j][i];
216         }
217     }
218 }
219 void Encrypt(int plaintext[4][4], int key[4][4]) // 加密
220 {
221     int en_or_de = 1;
222     int Exp[11][4][4];
223     KeyExpansion(key, Exp);
224     AddRoundKey(plaintext, Exp[0]);
225     for (int i = 1; i <= 10; ++i)
226     {
227         ByteSub(plaintext, en_or_de);
228         ShiftRow(plaintext, en_or_de);
229         if (i != 10)
230         {
231             MixColumn(plaintext, en_or_de);
232         }
233     }

```

```

233         AddRoundKey(plaintext , Exp[i]);
234     }
235 }
236 void Decrypt(int ciphertext[4][4] , int key[4][4]) //解密
237 {
238     int en_or_de = 0;
239     int Exp[11][4][4];
240     KeyExpansion(key , Exp);
241     AddRoundKey(ciphertext , Exp[10]);
242     for (int i = 9; i >= 0; --i)
243     {
244         ShiftRow(ciphertext , en_or_de);
245         ByteSub(ciphertext , en_or_de);
246         AddRoundKey(ciphertext , Exp[i]);
247         if (i != 0)
248         {
249             MixColumn(ciphertext , en_or_de);
250         }
251     }
252 }
253
254 int main() {
255     int key0[4][4] , plaintext0[4][4] , ciphertext0[4][4]; //记录第一组数
256     for (int k = 0; k < 2; k++)
257     {
258         int key[4][4] , plaintext[4][4] , ciphertext[4][4];
259         //cout << "输入密文(hex 128bit):";
260         for (int i = 0; i < 4; i++)
261         {
262             for (int j = 0; j < 4; j++)
263             {
264                 plaintext[j][i] = cases[k].plaintext[i][j];
265                 ciphertext[j][i] = cases[k].ciphertext[i][j];
266                 key[j][i] = cases[k].key[i][j];
267
268                 plaintext0[j][i] = cases[0].plaintext[i][j];

```

```

269         ciphertext0[j][i] = cases[0].ciphertext[i][j];
270         key0[j][i] = cases[0].key[i][j];
271     }
272 }
273 Encrypt(plaintext, key); //检查加密结果
274 bool t = true; //判断加密结果与样例结果是否一致
275 for (int i = 0; i < 4; i++)
276 {
277     for (int j = 0; j < 4; j++)
278     {
279         if (plaintext[j][i] != cases[k].ciphertext[i][j])
280         {
281             cout << "第" << k << "组，加密失败" << endl;
282             t = false;
283             break;
284         }
285     }
286 }
287 if (t)
288     cout << "第" << k << "组，加密成功" << endl;
289 Decrypt(ciphertext, key); //检查解密结果
290 bool t1 = true; //判断解密结果是否与样例一致
291 for (int i = 0; i < 4; i++)
292 {
293     for (int j = 0; j < 4; j++)
294     {
295         if (ciphertext[j][i] != cases[k].plaintext[i][j])
296         {
297             cout << "第" << k << "组，解密失败" << endl;
298             t1 = false;
299             break;
300         }
301     }
302 }
303 if (t1)
304     cout << "第" << k << "组，解密成功" << endl << endl;

```

```
305     }
306
307     //雪崩效应检测
308     int test_plaintext[4][4];
309     int test_old[4][4];
310     int test_key[4][4];
311     for (int i = 0; i < 4; i++)
312     {
313         for (int j = 0; j < 4; j++)
314         {
315             test_plaintext[j][i] = cases[0].plaintext[i][j];
316             test_old[j][i] = cases[0].plaintext[i][j];
317             test_key[j][i] = cases[0].key[i][j];
318         }
319     }
320     cout << endl;
321
322     string result;
323     cout << "改变明文" << endl;
324     int num1 = 0;
325     for (int i = 0; i <= 7; i++) {
326         result = int2binstr(test_plaintext);
327         if (result[i] == '0')
328             result[i] = '1';
329         else
330             result[i] = '0';
331         binstr2int(test_plaintext, result);
332         Encrypt(test_plaintext, key0);
333         int result_text = 0;
334         string s_new = int2binstr(test_plaintext);
335         string s = int2binstr(ciphertext0);
336         for (int i = 0; i < 128; i++) {
337             if (s_new[i] != s[i]) {
338                 result_text++;
339             }
340         }
```

```
341         cout << "change plaintext[" << i << "]: " << result_text << endl;
342         num1 += result_text;
343     }
344     cout << "average: " << (float)num1 / 8 << endl << endl;
345     cout << "改变密钥" << endl;
346     int num2 = 0;
347     for (int i = 0; i <= 7; i++) {
348         result = int2binstr(test_key);
349         if (result[i] == '0')
350             result[i] = '1';
351         else
352             result[i] = '0';
353         binstr2int(test_key, result);
354         Encrypt(test_old, test_key);
355         int result_text = 0;
356         string s_new = int2binstr(test_old);
357         string s = int2binstr(ciphertext0);
358         for (int i = 0; i < 128; i++) {
359             if (s_new[i] != s[i]) {
360                 result_text++;
361             }
362         }
363         cout << "change key[" << i << "]: " << result_text << endl;
364         num2 += result_text;
365     }
366     cout << "average: " << (float)num2 / 8 << endl;
367     return 0;
368 }
```