信息安全数学基础

实验指导书

教师：韩琦教授

实验地点：格物楼 213

实验时间：2023.11.25

一、背景介绍

高级加密标准（Advaneed Encryption Standard, AES）由比利时著名密码学家 Joan Daemen 和 Vincent Rijmen 设计, 是美因联邦政府采用的一种分组加密标准, 用来替代之前的 DES 算法，目前为对称密钥加密中最流行的算法之一。

本次实验的目的包括：

1. 理解和掌握 AES 的概念，以及 AES 在计算机加密通信领域的应用。
2. 了解 AES 加密解密的流程，将其和 DES 进行对比。
3. 编码实践 AES-128 加密解密的流程。

二、实验环境

- 操作系统：Windows7、Windows10、Windows11
- 编程语言：Java
- 代码编辑器：Eclipse

Ps：同学可以选择自己常用的代码编辑器。

三、实验内容

3.1 AES 整体介绍

AFS 算法的整体结构为代替一置换网络（Substitution-Permutation Network, SPN）结构，算法的明文长度为 128 位，密钥长度为 128、192 或 256 位，分别被称为 AES-128、AES-192 和 AES-256。AES 算法的整体结构如图 5-1 所示。

AES 的处理单位是字节，128 位的明文分组 P 和密钥 K 被分成 16 字节（以 128 位的密钥为例），分别记为 $P=P_0P_1\cdots P_{15}$；和 $K=K_0K_1\cdots K_{15}$。一般情况下，明文分组用以字节为单位的 4x4 矩阵描述，称为状态矩阵。在算法的每一轮中，状态矩阵的内容不断变化，最后结果作为密文输出。

AES 加密算法涉及 4 种操作：字节代替（SubByte）、行移位（ShifRow）、列混淆（MixColum）和轮密钥加（AddRoundKey）。在 AES-128 的加密算法中，共执行 10 次轮函数，前 9 次操作一样，包含以上 4 种操作，第 10 次没有列混淆。解密算法的每一步分别对应加密算法的逆操作，即解密时只需按照与加密时相反的顺序执行操作即可。除此之外，AES 算法还支持一种与加密顺序相同的解密方式，本书不再赘述，感兴趣的同学可以自行查阅相关资料了解。加、解密中每轮的密钥分别由种子密钥经过密钥扩展算法得到。算法中 16 字节的明文、密文和轮密钥都以一个 4 × 4 的字节矩阵表示**（注意：本实验中的明文、密文和密钥的 4*4 矩阵都是按列存储的，并非按行存储的！！！）**。

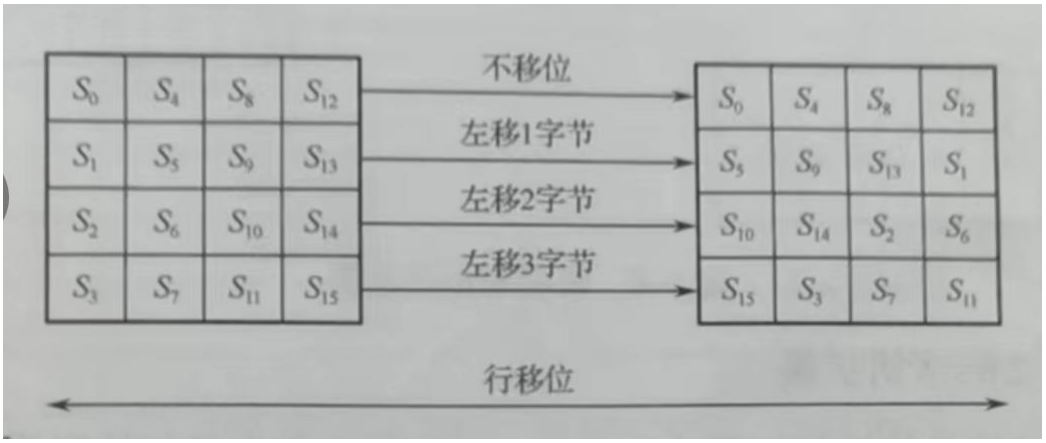
3.2 字节代替

字节代替的主要功能是通过 S 盒完成从一个字节到另一字节的映射。

S 盒和逆 S 盒分别为 16x16 的矩阵，完成一个从 8 位输入到 8 位输出的映射，输入的高 4 位对应的值作为行号，低 4 位对应的值作为列号。假设输入字节 $a=a_7a_6a_5a_4a_3a_2a_1a_0$，则输出的值为 $S[a_7a_6a_5a_4][a_3a_2a_1a_0]$，S 盒的逆变换同理。例如，字节 00000000B 变换后的值为 S[0][0] = 63H，再通过逆 S 盒即可得到变换前的值，$S^{-1}[6][3]$ = 00H。

3.3 行移位

行移位是一个简单的左循环移位操作。当密钥长度为 128 位时，状态矩阵的第 0 行左移 0 字节（不移位），第 1 行左移 1 字节，第 2 行左移 2 字节，第 3 行左移 3 字节。



3.4 列混淆

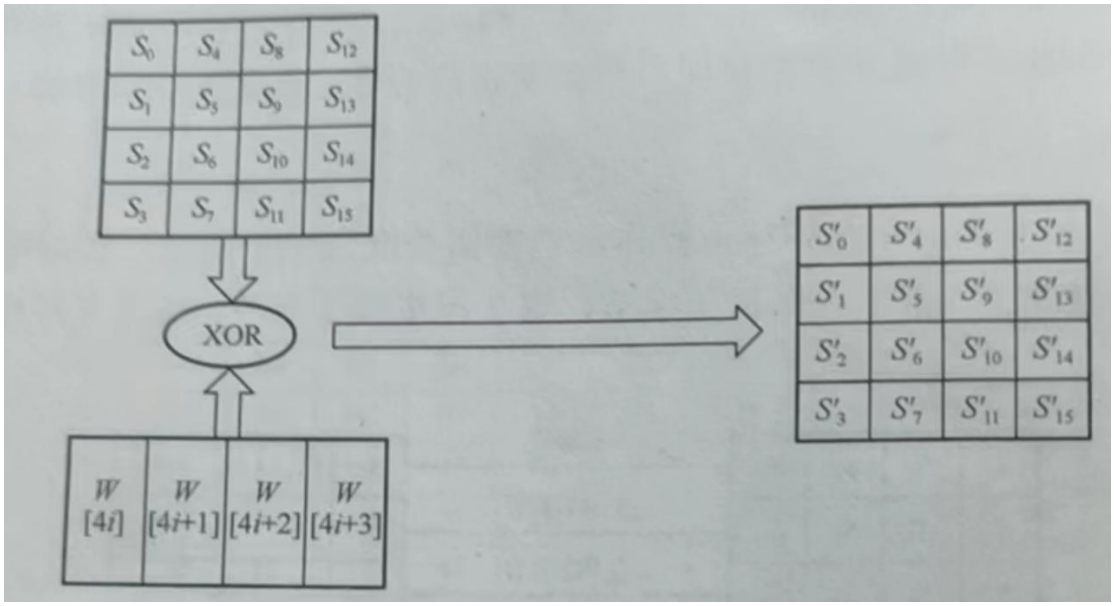
列混淆通过矩阵相乘实现，将经行移位后的状态矩阵与固定矩阵相乘，得到列混淆后的状态矩阵，其公式为

$$\begin{bmatrix} S'_{0,0} & S'_{0,1} & S'_{0,2} & S'_{0,3} \\ S'_{1,0} & S'_{1,1} & S'_{1,2} & S'_{1,3} \\ S'_{2,0} & S'_{2,1} & S'_{2,2} & S'_{2,3} \\ S'_{3,0} & S'_{3,1} & S'_{3,2} & S'_{3,3} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S'_{0,0} & S'_{0,1} & S'_{0,2} & S'_{0,3} \\ S'_{1,0} & S'_{1,1} & S'_{1,2} & S'_{1,3} \\ S'_{2,0} & S'_{2,1} & S'_{2,2} & S'_{2,3} \\ S'_{3,0} & S'_{3,1} & S'_{3,2} & S'_{3,3} \end{bmatrix}$$

其中，矩阵元素的乘法和加法都是定义在有限域 GF(2^8)上的二元运算，域的不可约多项式可以表示 0x11b。

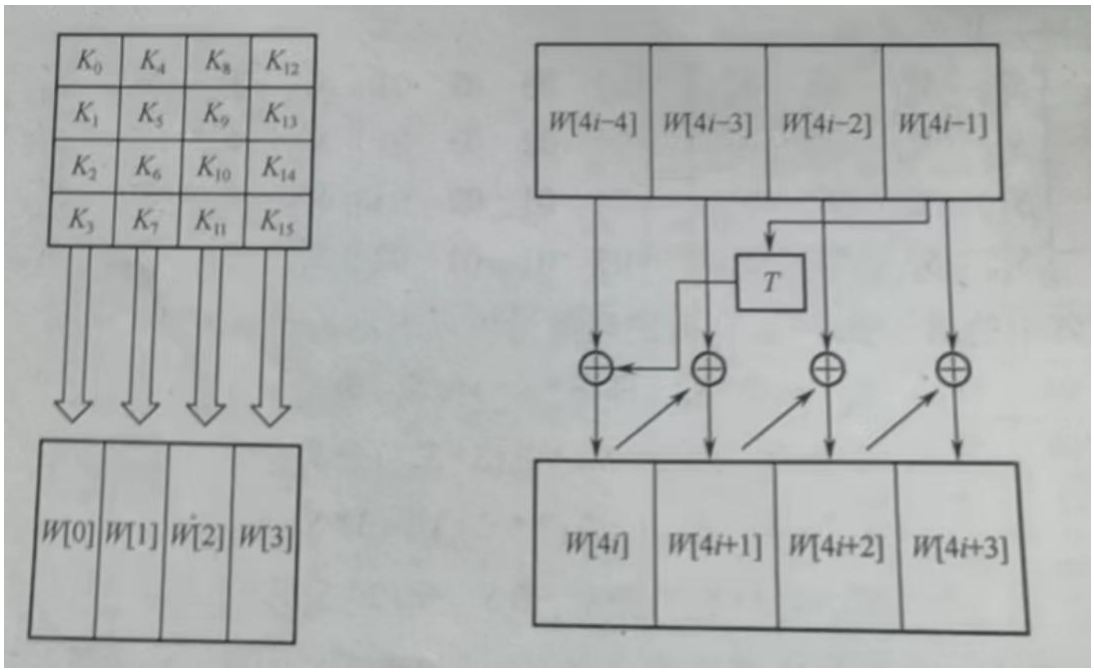
3.5 轮密钥加

轮密钥加将 128 位轮密钥 Ki 与状态矩阵中的数据进行逐位异或操作，如图所示。其中，密钥 Ki 共包含 4 个字（W[4i]，W[4i + 1]，W[4i + 2]，W[4i + 3]），每个字为 32 位（4 字节）。轮密钥加过程可看成按字逐位异或的过程，也可看成字节级别或比特级别的操作过程，即由 S0 S1 S2 S3 组成的 32 位与 W[4i]的异或运算。



3.6 AES 算法的密钥扩展

密钥扩展算法的具体过程如图所示。AES 算法首先将初始密钥输入一个 4×4 的状态矩阵。这个状态矩阵每一列的 4 字节组成一个字，将状态矩阵 4 列的 4 个字依次命名为 W[0]、W[1]、W[2]和 W[3]，它们构成一个以字为单位的数组。接着，对数组 W 扩充 40 个新列，构成 44 列的扩展密钥数组。



新列以如下递归方式产生：

（1） 如果 i 不是 4 的倍数，那么第 i 列由如下公式确定：

$$W[i]=W[i-4]\oplus W[i-1]$$

(2) 如果 i 是 4 的倍数，那么第 i 列由如下公式确定：

$$W[i]=W[i-4]\oplus T(W[i-1])$$

其中，T 是一个函数，由三部分组成：字循环、字代换和轮常量异或，三部分的操作分别如下：

① 字循环：将 1 个字中国的 4 字节循环左移 1 字节，即将输入字[b0, b1, b2, b3]变换成[b1, b2, b3, b0]。

② 字代换：对字循环对结果使用 S 盒进行字代换。

③ 轮常量异或：将前两步的结果同轮常量 Rcon[j]进行异或，其中 j 代表轮数。轮常量 Rcon[j]是一个字，其值见表：

j	Rcon[j]	j	Rcon[j]
1	01 00 00 00	6	20 00 00 00
2	02 00 00 00	7	40 00 00 00
3	04 00 00 00	8	80 00 00 00
4	08 00 00 00	9	1b 00 00 00
5	10 00 00 00	10	36 00 00 00

请补全 AES.java 代码中的方法，最终实现上述的算法，并且将代码的运行进行演示。

本次实验需要用到实验三的 GF(2 ^ 8)的加法和乘法运算，同学们需要使用上次实验的代码。

四、测试用例

完成实验，并通过下面的测试数据：

Test	密钥	明文	密文
0	0x0F1571C947D9E859 0CB7ADD6AF7F6798	0x0123456789ABCDEF FEDCBA9876543210	0xFF0B844A0853BF7C 6934AB4364148FB9
1	0x3475BD76FA040B7 3F521FFCD9DE93F24	0x1B5E8B0F1BC78D23 8064826704830CDB	0xF3855216DDF401D4 D42C8002E686C6E7
2	0x2B24424B9FED5966 59842A4D0B007C61	0x41B267BC5905F0A3 CD691B3DDAEE149D	0xFBA4EC67020F1573 ED28B47D7286D298

下面仅给出 Test0 的加密中间结果，以供同学们调试：

encrypt(0x0189FE7623ABDC5445CDBA3267EF9810, 0x0F1571C947D9E8590CB7ADD6AF7F6798, 10)

---
roundKey:
round 0: 0x0F470CAF15D9B77F71E8AD67C959D698
round 1: 0xDC9B97389049FE8137DF7215B0E93FA7
round 2: 0xD249DEE6C9807EFF6BB4C6D3B75E61C6
round 3: 0xC08957B1AF2F51AEDF6BAD7E396706C0
round 4: 0x2CA5F2435C73228C650EA3DDF1969050
round 5: 0x58FD0F4C9DEECC4036389B46EB7DEDBD
round 6: 0x718C83CFC729E5A54C74EFA9C2BF52EF
round 7: 0x37BB38F7143DD87D93E708A148F7A54A
round 8: 0x48F3CB3C261BC3BE45A2AA0B20D77238
round 9: 0xFD0EC5F90D16D56B42E04A41CB1C6E56
round 10: 0xB4BA7F868E984D26F3135918524E2076
round 0:
state:0x0ECEF2D936726B2B34251755AEB64E88
round 1:
afterByteSub: 0xAB8B893505407FF1183FF0FCE44E2FC4
afterShiftRow: 0xAB8B8935407FF105F0FC183FC4E44E2F
afterMixColumn: 0xB9945775E48E165147209A3FC5D6F53B
state: 0x650FC04D74C7E8D070FFE82A753FCA9C
round 2:
afterByteSub: 0x4D76BAE392C69B7051169BE59D7574DE
afterShiftRow: 0x4D76BAE3C69B70929BE55116DE9D7574
afterMixColumn: 0x8E22DB12B2F2DC92DF80F7C12DC51E52
state: 0x5C6B05F47B72A26DB43431129A9B7F94
round 3:
afterByteSub: 0x4A7F6BBF21403A3C8D18C7C9B814D222
afterShiftRow: 0x4A7F6BBF403A3C21C7C98D1822B814D2
afterMixColumn: 0xB1C10BCCBAF38B07F91F6AC31D19245C
state: 0x71485C7D15DCDAA92674C7BD247E229C
round 4:
afterByteSub: 0xA3524AFF598657D3F792C67A36F393DE
afterShiftRow: 0xA3524AFF8657D359C67AF792DE36F393
afterMixColumn: 0xD411FE0F3B440673CBAB623719B707EC
state: 0xF8B40C4C673724FFAEA5C1EAE82197BC
round 5:
afterByteSub: 0x418DFE29859A3616E40678879BFD8865
afterShiftRow: 0x418DFE299A3616857887E406659BFD88
afterMixColumn: 0x2A47C44883E818BA84182723EB100AF3
state: 0x72BACB041E06D4FAB220BC65006DE74E

round 6:

afterByteSub: 0x40F41FF2726F482D37B7654D633C942F

afterShiftRow: 0x40F41FF26F482D72654D37B72F633C94

afterMixColumn: 0x7B05424A1ED020409483185294C443FB

state: 0x0A89C185D9F9C5E5D8F7F7FB567B1114

round 7:

afterByteSub: 0x67A778973599A6D96168680FB12182FA

afterShiftRow: 0x67A7789799A6D935680F6168FAB12182

afterMixColumn: 0xEC1AC0800C5053C73BD700EFB72272E0

state: 0xDBA1F877186D8BBAA830084EFFD5D7AA

round 8:

afterByteSub: 0xB93241F5AD3C3DF4C204302F16030EAC

afterShiftRow: 0xB93241F53C3DF4AD302FC204AC16030E

afterMixColumn: 0xB11A44173D2FECB60A6B2F429F68F3B1

state: 0xF9E98F2B1B342F084FC98549BFBF8189

round 9:

afterByteSub: 0x991E73F1AF18153084DD973B08080CA7

afterShiftRow: 0x991E73F1181530AF973B84DDA708080C

afterMixColumn: 0x31303AC2AC718CC4466548EB6A1C3162

state: 0xCC3EFF3BA16759AF048502AAA1005F34

round 10:

afterByteSub: 0x4BB216E23285CB79F29777AC3263CF18

afterShiftRow: 0x4BB216E285CB793277ACF297183263CF

ciphertext: 0xFF0869640B53341484BFAB8F4A7C43B9

完成实验后将实验结果存放在实验报告中，并且和实验源码一起打包发送给助教，提交时**注明姓名和学号**。