VE475 Homework 3

Yiwen Yang

Ex.1 - Finite Fields

1. X^2+1 is of degree 2, so all possible factors should be of degree 1, which are X, $X+1$, $X+2$ in $\mathbb{F}_3[X]$.

   Try

$$\begin{aligned} X \cdot X &= X^2 \\ (X+1)(X+1) &= X^2+2X+1 \\ (X+2)(X+2) &= X^2+4X+4 = X^2+X+1 \\ X(X+1) &= X^2+X \\ X(X+2) &= X^2+2X \\ (X+1)(X+2) &= X^2+3X+2 = X^2+2 \end{aligned}$$

   None of the above polynomials equal to X^2+1, so X^2+1 is irreducible.

2. Let $P(X) = X^2+1$, $A(X) = 1+2X$, and there exists another two different $B(X)$ and $C(X)$ such that

$$A(X)B(X) \equiv A(X)C(X) \bmod P(X)$$

   Then $A(X)(B(X)-C(X)) \equiv 0 \bmod P(X)$, but $P(X)$ is irreducible in $\mathbb{F}_3[X]$, so it contradicts. This implies that there exists $B(X)$ such that

$$A(X)B(X) \equiv 1 \bmod P(X)$$

   And $B(X)$ is the multiplicative inverse of $A(X)$ modulo $P(X)$. So multiplicative inverse of $1+2X$ modulo X^2+1 exists in $\mathbb{F}_3[X]$.

3.

$$\begin{aligned}
X^2+1 &= (1+2X)2X+(-2X+1)\\
1+2X &= (-2X+1)(-1)+2\\
-2X+1 &= 2(-X)+1\\
1 &= (-2X+1)+2X\\
&= (-2X+1)+((1+2X)+(-2X+1))X\\
&= (1+2X)X+(X+1)(-2X+1)\\
&= (1+2X)X+(X+1)((X^2+1)-(1+2X)2X)\\
&= (X+1)(X^2+1)+(-2X^2-X)(1+2X)\\
&= (X+1)(X^2+1)+(X^2+2X)(1+2X)
\end{aligned}$$

So the multiplicative inverse of $1+2X$ modulo X^2+1 in $\mathbb{F}_3[X]$ is X^2+2X.

Ex.2 - AES

1. (a) Cyclically shift to the right row i by offset i, $0 \leq i \leq 3$.

   (b) The same as the encryption process, xor each byte with a byte of round key.

   (c) The MixColumns Layer is calculated by $C(X) \cdot X$, then to reverse it simply calculate $C^{-1}(X)$ multiplies the matrix. Multiply $C(X)$ by the given InvMixColumns yields 1, so InvMixColumns is $C^{-1}(X)$.

2. First AddRoundKey to the cipher text, then repeat (InvShiftRows, InvSubBytes, AddRoundKey, InvMixColumns) for 9 round, and for round 10, apply (InvShiftRows, InvSubBytes, AddRoundKey) to get the plain text.

3. Since InvShiftRows simply shifts the row position without changing values, InvSubBytes substitutes values without changing positions, the two steps may be reversed without changing the result.

4. (a) AddRoundKey and InvMixColumns both overwrites the cell value of the matrix, so reversing the two processes may lead to different results.

   (b)

$$e_{i,j} = (m_{i,j}a_{i,j}) \oplus k_{i,j}$$

   (c)

$$\begin{aligned}
m_{i,j}a_{i,j} &= e_{i,j} \oplus k_{i,j}\\
a_{i,j} &= m_{i,j}^{-1}(e_{i,j} \oplus k_{i,j})\\
&= (m_{i,j}^{-1}e_{i,j}) \oplus (m_{i,j}^{-1}k_{i,j})
\end{aligned}$$

(d) When generating inverse round keys, left multiply the round keys with the inverse mix columns, *i.e.*, $k'_{i,j} = m^{-1}_{i,j} k_{i,j}$. Then InvAddRoundKey simply xor the matrix generated from InvMixColumns with the inverse round keys.

5. First AddRoundKey to the cipher text, then repeat (InvShiftRows, InvSubBytes, InvMixColumns, InvAddRoundKey) for 9 round, and for round 10, apply (InvShiftRows, InvSubBytes, AddRoundKey) to get the plain text.

6. The decryption process is as the same structure of the encryption process, only needed to implemented the inverse functions to replace into the layers without fixing the layer orders.

Ex.3 - DES

1. DES encrypts and decrypts block cipher based on Feistel Network. Encryption uses a 64-bit key and the block size as 64-bit, decryption uses the same key and the cipher text block is also 64-bit. The key is 64-bit in length, but every 8-bit there is a odd parity bit, *i.e.*, the key is composed of a 56-bit key and 8 parity bits. The 64-bit is processed by key schedule (KS) to generate 16 sub keys $K_1 \sim K_{16}$, each of 48-bit.

   The general process of DES goes through as

   (a) Apply Initial Permutation (IP) on the input (plain text). IP uses the Table 1 to replace the bits. After IP, output should be input[58], input[50], ..., input[7].

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

Table 1: Initial Permutation

   (b) Take the left 32 bits of output from IP as L_0, and right 32 bits as R_0. Feed (L_0, R_0, function F, sub key K_1) into Feistel Network, *i.e.*, output ($L_1 = R_0$, $R_1 = L_0 \oplus F(R_0, K_1)$).

   (c) Repeat Step 1b for 16 times. For the 16th time, switch the left and right part of output, *i.e.*, ($L_{16} = L_{15} \oplus F(R_{15}, K_{16})$, $R_{16} = R_{15}$).

   (d) For the final 64-bit output of Feistel Network, apply Inverse Initial Permutation (IP-1) to get the cipher text. IP-1 is to replace bits with Table 2.

   Following is detail about function F and key schedule (KS).

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

Table 2: Inverse Initial Permutation

- **Function** F

  F takes 32-bit R and 48-bit K as input, yields a 32-bit output.

  (a) Use function E to get a 48-bit output from R. E is to replace bits by Table 3.

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

Table 3: function E

  (b) Xor the 48-bit result with K.

  (c) Split the 48-bit result by 6 bits to get 8 partitions, and apply functions S_1, S_2, ..., S_8 in order. The output of each S function is 4 bits, so the overall output will be 32 bits.
  Here takes S_1 as an example of function S. All 8 S functions are listed in Table 6.
  S_{in} is the 6-bit input of S, use Table 4 to transform bits.
  The transform rule is defined as

    i. Row index is $\{S_{in}[0], S_{in}[5]\}$.

    ii. Column index is $S_{in}[1:4]$.

    iii. Find corresponding number (4-bit) in the table as the output.

  (d) Apply P on the 32-bit output from the last step, and get the 32-bit output to be the output of function F. P is defined to replace bits using Table 5.

- **Key Schedule (KS)**

  KS gets the 64-bit key as input and outputs 16 48-bit sub keys.

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Table 4: S-box S_1

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

Table 5: function P

(a) Transform bits of the key using Permuted Choice 1 (PC-1). Apply Table 7 to get 28-bit C_0, and Table 8 to get 28-bit D_0.

(b) Rotate Shift C_0 and D_0 to the left to yield C_1 and C_2. The shift bit should be either 1 or 2, which is specified in Table 9.

(c) Apply Permuted Choice 2 (PC-2) on C_1 and D_1. PC-2 transforms $\{C_1, D_1\}$ (56-bit) to a 48-bit sub key K_1. The transform table is shown in Table 10.

(d) Repeat Step 1b to Step 1c for 16 times, yielding sub keys $K_1 \sim K_{16}$.

2.
- Linear cryptanalysis tries to discover affine relations between cipher text and plain text to detect key bits.
- Differential cryptanalysis studies how different plain text or cipher text affects output result so as to figure out some critical

3. Double DES uses two instance of DES with two different keys, and triple DES use three instance with three different keys. Double DES is vulnerable since its security level is not 2^{112} with 112-bit key, but instead 2^{56} if meet-in-the-middle attack is used, which is very easy to break with modern technology. Triple DES has the security level of 2^{112}, which can satisfy modern security criterion.

4. Passwords stored with `passwd` on Unix systems are encrypted with DES by default, which is not safe since DES is vulnerable and easy to break today. Better solutions would be to use hash functions such as SHA256, because there is no need to decrypt a password but only a verification is required.

S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Table 6: S-box $S_2 \sim S_8$

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36

Table 7: Permuted Choice 1 C

63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Table 8: Permuted Choice 1 D

Shift Times	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Shift bits	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Table 9: Rotate Shift

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Table 10: Permuted Choice 2

Ex.4 - Programming

See in folder **ex4**.