VE475 Homework 3

Liu Yihao 515370910207

Ex. 1 — Finite fields

1. The possible factors of X^2+1 in $F_3[X]$ are $X, X+1, X+2$

$$X(X+1) = X^2+X \neq X^2+1$$
$$X(X+2) = X^2+2X \neq X^2+1$$
$$(X+1)(X+2) = X^2+3X+2 = X^2+2 \neq X^2+1$$
$$X \cdot X = X^2 \neq X^2+1$$
$$(X+1)(X+1) = X^2+2X+1 \neq X^2+1$$
$$(X+2)(X+2) = X^2+4X+4 = X^2+X+1 \neq X^2+1$$

   So X^2+1 is irreducible in $F_3[X]$

2. According to the Proof on c2, Page 39, if $P(X)$ is irreducible and $A(X)$ is a polynomial in a finite field, there exists a polynomial $B(X)$ such that

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

   Here let $P(X) = X^2+1$, $A(X) = 1+2X$, then $B(X)$ is the multiplication inverse of $1+2X \bmod X^2+1$.

3. Applying the Extended Euclid Algorithm,

	q_i	r_i	s_i	t_i
0		$2X+1$	1	0
1		X^2+1	0	1
2	$(2X+1) \div (X^2+1) = 0$	$2X+1$	1	0
3	$(X^2+1) \div (2X+1) = 2X$	$X+1$	X	1
4	$(2X+1) \div (X+1) = 2$	2	$X+1$	1
5	$(X+1) \div 2 = 2X$	1	X^2+2X	$X+1$

$$(1+2X)(X^2+2X) \equiv 1 \bmod X^2+1$$

Ex. 2 — AES

1. (a) *InvShiftRows* cyclically shift to the right row i by offset i, $0 \leqslant i \leqslant 3$.

   (b) The inverse of *AddRoundKey* is actually the same as itself, since if we xor a value by another value twice, it will keep not changed. We only need to reverse the order of round keys.

   (c) The transformation matrix of *MixColumns* is

$$A = \begin{pmatrix} 00000010 & 00000011 & 00000001 & 00000001 \\ 00000001 & 00000010 & 00000011 & 00000001 \\ 00000001 & 00000001 & 00000010 & 00000011 \\ 00000011 & 00000001 & 00000001 & 00000010 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$$

   If the transformation matrix of *InvMixColumns* is

$$B = \begin{pmatrix} 00001110 & 00001011 & 00001101 & 00001001 \\ 00001001 & 00001110 & 00001011 & 00001101 \\ 00001101 & 00001001 & 00001110 & 00001011 \\ 00001011 & 00001101 & 00001001 & 00001110 \end{pmatrix} = \begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix}$$

   We can calculate BA according to the definition of $GF(2^8)$.
   For example (in hex form), in the first column,

$$\begin{aligned} (0E \cdot 02) \oplus (0B \cdot 01) \oplus (0D \cdot 01) \oplus (09 \cdot 03) &= 01 \\ (09 \cdot 02) \oplus (0E \cdot 01) \oplus (0B \cdot 01) \oplus (0D \cdot 03) &= 00 \\ (0D \cdot 02) \oplus (09 \cdot 01) \oplus (0E \cdot 01) \oplus (0B \cdot 03) &= 00 \\ (0B \cdot 02) \oplus (0D \cdot 01) \oplus (09 \cdot 01) \oplus (0E \cdot 03) &= 00 \end{aligned}$$

   The calculation of other three column is similar, thus we can get

$$BA = \begin{pmatrix} 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 00 & 00 & 01 & 00 \\ 00 & 00 & 00 & 01 \end{pmatrix} = \begin{pmatrix} 00000001 & 00000000 & 00000000 & 00000000 \\ 00000000 & 00000001 & 00000000 & 00000000 \\ 00000000 & 00000000 & 00000001 & 00000000 \\ 00000000 & 00000000 & 00000000 & 00000001 \end{pmatrix} = I$$

   If the origin matrix is S, the mix-columned matrix is AS, then

$$B(AS) = BA(S) = IS = S$$

2. First, we generate the round keys according to the key, then we apply *AddRoundKey* with round key (40–43).

   Second, we apply nine turns of following four steps (i is the turn number): *InvShiftRows*, *InvSubBytes*, *AddRoundKey* with round key ($40 - 4 * i$–$43 - 4 * i$) and *InvMixColumns*.

   At last, we apply *InvShiftRows*, *InvSubBytes* and *AddRoundKey* with round key (0–3).

3. Since *InvShiftRows* doesn't change the value of any cell, and *InvSubBytes* only substitutes the value of each cell according to a table, the order of applying them doesn't influence the result. So they can be applied on reverse order.

4. (a) Since *InvMixColumns* and *AddRoundKey* affect the value of each column based on completely different theorems, the reverse order may cause a different result.

(b)

$$[(m_{i,j})(a_{i,j})] \oplus (k_{i,j})$$

(c)

$$(a_{i,j}) = (m_{i,j})^{-1}[(e_{i,j}) \oplus (k_{i,j})] = [(m_{i,j})^{-1}(e_{i,j})] \oplus [(m_{i,j})^{-1}(k_{i,j})]$$

So the inverse operation is given by

$$(e_{i,j}) \longrightarrow (m_{i,j})^{-1}(e_{i,j}) \oplus (m_{i,j})^{-1}(k_{i,j})$$

(d) *InvAddRoundKey* first apply *InvMixColumns* to the key, then apply *AddRoundKey* to the data with the inv-mix-columned key.

5. First, we generate the round keys according to the key, then we apply *AddRoundKey* with round key (40–43).

   Second, we apply nine turns of following four steps (i is the turn number): *InvSubBytes*, *InvShiftRows*, *InvMixColumns* and *InvAddRoundKey* with round key ($40 - 4 * i$–$43 - 4 * i$).

   At last, we apply *InvSubBytes*, *InvShiftRows* and *AddRoundKey* with round key (0–3).

6. The advantage of this strategy is that we only need to implement the four inverse transformations and apply them in the same order as the encryption process. Thus the encryption and decryption process can be unified and written only once.

Ex. 3 — DES

1. In DES, the length of plaintext is 64 bits, and the length of key is 56 bits.

   First, the plaintext is transformed according to the following table:

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

   Then the transformed plaintext is divided into two parts of 32 bits, and we apply Feistel Network for 16 turns, each turn can be expressed as

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

   The procedure of function $F(R_{i-1}, K_i)$ is

   (a) Expand R_{i-1} (32 bits) into 48 bits a according to the following table:

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

(b) Xor the 48 bits data and the key K_i

(c) Reduce the 48 bits data into 32 bits with eight S-boxes. Each S-box accept 6 bits and output 4 bits.

S_1	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
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

(d) At last, we apply another transformation to the 32 bits data according to the following table:

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

The round keys K_i are generated in this method:

(a) First, transform the 56 bits key K according to the following table and divide the result into two 28 bits data C_0 and D_0.

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

(b) Left shift C_{i-1} and D_{i-1} according to the following table to get C_i and D_i, where $1 \leqslant i \leqslant 16$.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
shift	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

(c) Concat 28 bits C_i and D_i and transform it according to the following table, which forms 48 bits K_i.

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

At last, we apply a reverse transformation, which is shown in the following table.

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

The decryption method is actually the same as the encryption method, we need only reverse the order of K_i.

2. Linear cryptanalysis:

There are two parts to linear cryptanalysis. The first is to construct linear equations relating plaintext, ciphertext and key bits whose probabilities of holding are close to 0 or 1. The second is to use these linear equations in conjunction with known plaintext-ciphertext pairs to derive key bits.

Differential cryptanalysis:

The attack discover whether a given input/output difference pattern only occurs for certain values of inputs. Observing the desired output difference (between two chosen or known plaintext inputs) suggests possible key values.

3. The safety of double DES is similar to single DES, since it suffers meet-in-the-middle attack. suppose the encryption map relation is $X \to E(K, X)$, the decryption map relation is $X \to D(K, X)$, the plaintext is P, the ciphertext is C, then

$$C = E(K_2, E(K_1, P))$$

Suppose the attacker knows a pair of P and C, $\exists X$ such that

$$X = E(K_1, P) = D(K_2, C)$$

First, he can try to encrypt P with all of 2^{56} values of K_1 into 2^{56} values of X_1. Then he can decode C with all of 2^{56} values of K_2 into X_2 until he find that X_2 is identical to one of X_1. In this situation, there are 2^{112} possible combinations of K_1, K_2, so there are $2^{112}/2^{64} = 2^{48}$ wrong results.

However, if the attacker knows two pairs of P and C, the possibility of wrong result decreases to $2^{48}/2^{64} = 2^{-16}$, which is very small. So we can concluded that if a key pair K_1, K_2 satisfy both two pairs of P and C, they are the actual keys.

With this method, a double DES can be broken in 2^{57} operations, which is not considered safe anymore in modern cryptography.

If we use triple DES, we can avoid this kind of attack. Suppose we also use two keys K_1 and K_2 to apply the encryption and decryption procedure:

$$C = E(K_1, D(K_2, E(K_1, P)))$$

$$P = D(K_1, E(K_2, D(K_1, P)))$$

People often use this kind of encryption and decryption method in triple DES because if $K_1 = K_2$, the algorithm behaves same as a single DES, which can provide compatibility.

If meet-in-the-middle is applied to triple DES, the time complexity becomes 2^{112} to encrypt, which can be considered safe anymore in modern cryptography.

4. According to the *man passwd*, The legacy UNIX System encryption method is based on the NBS DES algorithm. User's password will be encrypted and saved to */etc/passwd*. However, it is a problem that single DES is no longer safe now. So for security, modern Unix based system use more recent methods, such as SHA, to validate the password.

Ex. 4 — Programming

In the ex4 folder, with a README file inside it.