Problem 1. Finite fields

1. We could list all the possible factors of X^2+1 in $F_3[X]$, which are

$$0, 1, 2, X, X+1, X+2, 2X, 2X+1, 2X+2$$

Multiplying each other and we can show the following

$$\begin{aligned}
X(X) &= X^2 \\
X(X+1) &= X^2+X \\
X(X+2) &= X^2+2X \\
X(2X) &= 2X^2 \\
X(2X+1) &= 2X^2+X \\
X(2X+2) &= 2X^2+2X \\
(X+1)(X+1) &= X^2+2X+1 \\
(X+1)(X+2) &= X^2+3X+2 = X^2+2 \\
(X+1)(2X) &= 2X^2+2X \\
(X+1)(2X+1) &= 2X^2+3X+1 = 2X^2+1 \\
(X+1)(2X+2) &= 2X^2+3X+2 = 2X^2+2 \\
(X+2)(X+2) &= X^2+4X+4 = X^2+X+1 \\
(X+2)(2X) &= 2X^2+4X = 2X^2+X \\
(X+2)(2X+1) &= 2X^2+4X = 2X^2+X \\
(X+2)(2X+2) &= 2X^2+6X+4 = 2X^2+1 \\
(2X)(2X) &= 4X^2 = X^2 \\
(2X)(2X+1) &= 4X^2+2X = X^2+2X \\
(2X)(2X+2) &= 4X^2+4X = X^2+X \\
(2X+1)(2X+1) &= 4X^2+4X+1 = X^2+X+1 \\
(2X+1)(2X+2) &= 4X^2+6X+2 = X^2+2
\end{aligned}$$

Since the results above all $\neq X^2+1$, we can conclude that X^2+1 is irreducible in $F_3[X]$

2. Let $A(X), B(X)$ and $C(X)$ be three distinct non-zero polynomials such that

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

This implies $A(X)(B(X)-C(X)) \equiv 0 \mod P(X)$. If we let $P(X) = X^2+1$, this is not possible because X^2+1 is irreducible. For $A(X)$ to be a polynomial in a finite field, there exists a polynomial $B(X)$ such that

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

Let $A(X) = 1+2X$, then $B(X)$ is the multiplication inverse of $1+2X \mod X^2+1$.

3. As we have shown in the first question

$$(2X+1)(2X+2) = 4X^2 + 6X + 2 = X^2 + 2$$
$$X^2 + 2 \equiv 1 \mod X^2 + 1$$

Hence, we have found the multiplicative inverse is $2X + 2$

Problem 2. AES

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

   (b) Inverse of the layer AddRoundKey is the same as AddRoundKey, $A \oplus K = B \longrightarrow B \oplus K = A$

   (c) We know the transformation matrix of *MixColumns* is

$$C(X) = \begin{pmatrix} 00000010 & 00000011 & 00000001 & 00000001 \\ 00000001 & 00000010 & 00000011 & 00000001 \\ 00000001 & 00000001 & 00000010 & 00000011 \\ 00000011 & 00000001 & 00000001 & 00000010 \end{pmatrix}$$

For an input matrix X and output matrix Y, it satisfies

$$C \times X = Y$$
$$C^{-1} \times C \times X = Y \times C^{-1}$$
$$X = Y \times C^{-1}$$

We can verify that the transformation matrix of InvMixColumns is the given one, let's call it D, by checking if it times $C(X)$ equals the identity matrix

$$C \times D = \begin{pmatrix} 00000010 & 00000011 & 00000001 & 00000001 \\ 00000001 & 00000010 & 00000011 & 00000001 \\ 00000001 & 00000001 & 00000010 & 00000011 \\ 00000011 & 00000001 & 00000001 & 00000010 \end{pmatrix}$$
$$\times \begin{pmatrix} 00001110 & 00001011 & 00001101 & 00001001 \\ 00001001 & 00001110 & 00001011 & 00001101 \\ 00001101 & 00001001 & 00001110 & 00001011 \\ 00001011 & 00001101 & 00001001 & 00001110 \end{pmatrix}$$
$$= \begin{pmatrix} 00000001 & 00000000 & 00000000 & 00000000 \\ 00000000 & 00000001 & 00000000 & 00000000 \\ 00000000 & 00000000 & 00000001 & 00000000 \\ 00000000 & 00000000 & 00000000 & 00000001 \end{pmatrix} = I$$

Hence we can conclude that matrix $D = C^{-1}$, the given matrix is the transformation matrix of *InvMixColumns*.

2. First, we obtain the round key, then apply AddRoundKey with key $K[40-43]$.
   Next, we apply 9 rounds of the following operations in this listing order: *InvShiftRows*, *InvSubBytes*, *AddRoundKey*, *InvMixColumns*. For *AddRoundKey* in the i-th round, key is given by the $K[40-4i], ..., K[43-4i]$.
   Finally, we apply *InvShiftRows*, *InvSubBytes* and *AddRoundKey* with key $K[0], ..., K[3]$.

3. Because *InvShiftRows* only changes the position of some values in the matrix and it doesn't affect the values. *InvSubBytes* gives the same result in whatever order since it generates values through looking up the corresponding value in a table. Reverse those two operations obviously won't change the final result.

4. (a) It is because *AddRoundKey* is calculated by XOR and *InvMixColumns* is by matrix multiplication. The order matters here because it could affect the final result, thus it is not possible to reverse the order of *AddRoundKey* and *InvMixColumns*.

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

   (c) Consider $(e_{i,j})$ be the result applied by MixColumns and then AddRoundKey.

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

   (d) First we generate the key by applying the InvMixColumns to the key, then we apply the AddRoundKey using those generated key.

5. First, we obtain the round key, then apply AddRoundKey with key $K[40-43]$.
   Next, we apply 9 rounds of the following operations in this listing order: *InvSubBytes*,*InvShiftRows*, *InvMixColumns*, *InvAddRoundKey*. For *AddRoundKey* in the i-th round, key is given by the $K[40-4i], ..., K[43-4i]$.
   Finally, we apply *InvSubBytes*, *InvShiftRows* and *AddRoundKey* with key $K[0], ..., K[3]$.

6. The advantages of this strategy is that the decryption process has the same order of inverse transformation as the encryption process, and we don't need to change the whole structure.

Problem 3. DES

1. The Data Encryption Standard (DES) is a symmetric-key algorithm for the encryption of digital data. Plaintext has a size of 64 bits, and the key size is 56 bits. The encryption procedure includes initial Permutation (IP), final Permutation (FP) and 16 rounds of substitution and transposition. The detailed step listed below

   (a) Create 16 subkeys, each of which is 48-bits long.
   The 64-bit key is permuted according to the following table. Since the first entry

in the table is "58", this means that the 58th bit of the original key K becomes the first bit of the permuted key K+.

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
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

Figure 1: 64 to 56 bit permutation

After we get the 56-bit permutation from the 64 bit key, we split this key into left and right halves (C_0, D_0). And we will create sixteen pair of (C_N, D_N) by left shifting the (C_{N-1}, D_{N-1}) pair. We now form the keys Kn, for $1 \leq n \leq 16$, by applying the following permutation table to each of the concatenated pairs (Cn, Dn). From this 56-bit key, a different 48-bit Sub Key is generated.

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

Figure - compression permutation

(b) Encode each 64-bit block of data.
There is an initial permutation IP of the 64 bits of the message data M, which rearranges the bits according to the following table.

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	33	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Figure - Initial permutation table

Next, we divide the permuted block IP into a left half L0 of 32 bits, and a right half R0 of 32 bits. We now proceed through 16 iterations, for $1 \leq n \leq 16$, using a function f which operates on two blocks–a data block of 32 bits and a key Kn of 48 bits–to produce a block of 32 bits. In each iteration, we take the right 32 bits

of the previous result and make them the left 32 bits of the current step. For the right 32 bits in the current step, we XOR the left 32 bits of the previous step. Now we need to expand each block we previously get from 32 bits to 48 bits by using the following block.

E BIT-SELECTION TABLE

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

Figure 2: Expansion table

We now have 48 bits, or eight groups of six bits. We now do something strange with each group of six bits: we use them as addresses in tables called "S boxes". Each group of six bits will give us an address in a different S box. Located at that address will be a 4 bit number. This 4 bit number will replace the original 6 bits. The net result is that the eight groups of 6 bits are transformed into eight groups of 4 bits (the 4-bit outputs from the S boxes) for 32 bits total.

Finally, we apply a final permutation as defined by the following table.

IP^{-1}

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

Figure 3: Final permutation table

Decryption is simply the inverse of encryption, follwing the same steps as above, but reversing the order in which the subkeys are applied.

2. Linear cryptanalysis aims to construct a linear approximation on the action of the cipher. There are two parts to linear cryptanalysis. The first is to construct linear equations relating plaintext, ciphertext and key bits that have a high bias; that is, whose probabilities of holding (over the space of all possible values of their variables)

are as close as possible 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 is applicable primarily to block ciphers, and it is the study of how differences in information input can affect the resultant difference at the output.

3. Triple-DES runs through one instance of DES, feeds that output as input to another instance of DES and finally that output goes in to a third. The reason we use 3DES over 2DES is that because of meet-in-the-middle attack, the security level of 2DES using 112 key bits is about 2^57, and 3DES makes it need extra work on the third operation to tell if they met in the middle and thus we can't simply look for where the first and last operation produce the same value. Therefore it imporves much the security levels.

4. According to the *man passwd*, the legacy UNIX System encryption method is based on the NBS DES algorithm. The following path stores user's password info.

```
/etc/master.passwd  The user database
/etc/passwd         A Version 7 format password file
/etc/passwd.XXXXXX  Temporary copy of the password file
```

   Since single DES is not secure enough now, the modern UNIX systems should look for better encryption methods to secure the passwords.