Midterm

111550057 資工15 莊婷馨

Problem 1

a)

$$f(x)+g(x) = x^3+2x^2+2 \equiv x^3$$

$$f(x)-g(x) = -x^3 \equiv x^3$$

$$[f(x) \times g(x)] \bmod P(x)$$

$$= [(x^2+1)(x^3+x^2+1)] \bmod P(x)$$

$$= [x^5+x^4+x^3+2x^2+1] \bmod P(x)$$

$$\equiv [x^5+x^4+x^3+1] \bmod (x^4+x+1) \equiv x^3+x^2$$

$$\begin{array}{r l}
 & x \quad +1 \\
\cline{2-2}
x^4+x+1 \,\big) & x^5+x^4+x^3+0x^2+0x+1 \\
 & x^5 \qquad\qquad +x^2+x \\
\cline{2-2}
 & \quad x^4+x^3+x^2+x+1 \\
 & \quad x^4 \qquad\qquad +x+1 \\
\cline{2-2}
 & \quad x^3+x^2
\end{array}$$

b)

$$f(x)+g(x) = x^2+x+2 \equiv x^2+x$$

$$f(x)-g(x) = x^2-x \equiv x^2+x$$

$$[f(x) \times g(x)] \bmod P(x)$$

$$= [(x^2+1)(x+1)] \bmod P(x)$$

$$= [x^3+x^2+x+1] \bmod (x^4+x+1)$$

$$\equiv x^3+x^2+x+1$$

Problem 2

a)

$$f(x)+g(x) = x^7+x^5+x^4+x^3+2x+2$$
$$\equiv x^7+x^5+x^4+x^3$$
$$f(x)-g(x) = x^7+x^5+x^4-x^3$$
$$\equiv x^7+x^5+x^4+x^3$$

$$[f(x)\times g(x)] \bmod m(x)$$
$$[(x^7+x^5+x^4+x+1)(x^3+x+1)] \bmod m(x)$$
$$= [x^{10}+\cancel{x^8}+\cancel{x^7}+\cancel{x^8}+x^6+\cancel{x^5}+\cancel{x^7}+\cancel{x^5}+\cancel{x^4}+\cancel{x^4}+x^2+\cancel{x}+x^3+\cancel{x}+1]$$
$$\bmod m(x)$$
$$\equiv [x^{10}+x^6+x^3+x^2+1] \bmod (x^8+x^4+x^3+x+1)$$
$$\equiv x^5+1$$

b)

$$(x+1)^4$$
$$=(x^2+1)(x^2+1)$$
$$= x^4+1$$
$\therefore$ x^4+1 is reducible over $GF(2^8)$

Problem 3

a)

$$f(x)\times g(x) \equiv 1 \pmod{P(x)}$$
$$x\cdot g(x) \equiv 1 \pmod{(x^4+x+1)}$$
$$g(x) = x^3+1$$

b)

$$g(x)=x^2+x$$

$f(x)=1$	$\times$
$f(x)=x$	$\times$
$f(x)=x+1$	$\times$
$f(x)=x^2$	$\times$

$f(x) = x^2+1 \qquad f(x) \times g(x) = x^4+x^3+x^2+x$ ✗

$f(x) = x^2+x+1 \qquad f(x) \times g(x) \equiv x^4+x$ ✓

$x^4+x \equiv 1 \ (\text{mod } (x^4+x+1))$

$f(x) = x^2+x+1$ ※

Problem 4

a)

Calculate the inverse of x^3+x^2, $g(x)$:

$f(x) = x^3+x^2$, $P(x) = x^8+x^4+1$

Let $f(x)g(x) \equiv 1 \ (\text{mod } P(x))$

$P(x) = (x^5+x^4+x^3+x^2) \cdot f(x) + 1$

$g(x) = x^5+x^4+x^3+x^2$

$\Rightarrow (x^3+x^2+x)\ (x^3+x^2)^{-1} \ \text{mod } (x^8+x^4+1)$

$= (x^3+x^2+x)\ (x^5+x^4+x^3+x^2) \ \text{mod } (x^8+x^4+1)$

$= (x^8+x^6+x^5+x^3) \ \text{mod } (x^8+x^4+1)$

$= x^6+x^5+x^4+x^3+1$ ※

b)

$(x^6+x^3+1)\ (x^4+x+1) \ \text{mod } (x^8+x^4+1)$

$= (x^{10}+x^9+x^7+x^4+1) \ \text{mod } (x^8+x^4+1)$

$= x^7+x^6+x^5+x^4+x^2+x+1$ ※

$$
\begin{array}{r|l}
 & x^2+x \\
\hline
x^8+x^4+1 & x^{10}+x^9+x^7+x^4+1 \\
 & x^{10}+x^6+x^2 \\
\hline
 & x^9+x^7+x^6+x^4+x^2+1 \\
 & x^9+x^5+x \\
\hline
 & x^7+x^6 x^5+x^4+x^2+x+1
\end{array}
$$

Problem 5

a)

Since the calculation of mix column performs as follows

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$\begin{aligned} d_0 &= 2 \bullet b_0 \oplus 3 \bullet b_1 \oplus 1 \bullet b_2 \oplus 1 \bullet b_3 \\ d_1 &= 1 \bullet b_0 \oplus 2 \bullet b_1 \oplus 3 \bullet b_2 \oplus 1 \bullet b_3 \\ d_2 &= 1 \bullet b_0 \oplus 1 \bullet b_1 \oplus 2 \bullet b_2 \oplus 3 \bullet b_3 \\ d_3 &= 3 \bullet b_0 \oplus 1 \bullet b_1 \oplus 1 \bullet b_2 \oplus 2 \bullet b_3 \end{aligned}$$

It is possible to pre-compute the results of the multiplication of input (b_ns) and 1, 2, or 3. There are only 256 possible value for every b_n. Then do XOR on the values to get the result (d_ns).

b)

For byte substitution, it is performed by looking up substitution box. For shift rows, it shifts the state matrix by a certain number of positions (from 0 to 3). Both operations has a limited number of possible outcomes with the 256 kinds of possible inputs.

As for the mix-column operation, we can expand the table by listing all possible outcomes after XORing the four results.

Then the three operations are possible to be implemented as a simple look-up table operation by combining the three look-up tables.

Problem 6

a)

For implementation with mix-columns operation in the last round:

Encryption:

```
Round(State, ExpandedKey[i]) {
    SubBytes(State);
    ShiftRows(State);
    MixColumns(State);
    AddRoundKey(State, ExpandedKey[i]);
}
```

Decryption:

```
InvRound(State, ExpandedKey[i]) {
    AddRoundKey(State, ExpandedKey[i]);
    InvMixColumns(State);
    InvShiftRows(State);
    InvSubBytes(State);
}
```

Together the whole process of encryption and decryption:

```
AddRoundKey(ExpandedKey[0])
for i = 1 .. 10:
    Round(State, ExpandedKey[i])

for i = 10 .. 1:
    InvRound(State, ExpandedKey[i])
AddRoundKey(ExpandedKey[0])
```

Clearly, the “Round” and “InvRound” has different sequences.

To make the two processes have a similar form, first swap InvSubBytes and InvShiftRows. Since their order is indifferent, it does not effect the process.

Then swap the order of AddRoundKey and InvMixColumns. To get the same outcome, we have to modify the key by doing mix-column operation to them:

```
InvRound(State, ExpandedKey[i]) {
    InvMixColumns(State);
    AddRoundKey(State, InvMixColumns(ExpandedKey[i]));
    InvSubBytes(State);
    InvShiftRows(State);
}
```

Since there is no mix-columns operation in the last round, the new InvRound can be expressed as:

```
InvRound(State, ExpandedKey[i]) {
    InvSubBytes(State);
    InvShiftRows(State);
    InvMixColumns(State);
    AddRoundKey(State, InvMixColumns(ExpandedKey[i]));
}
```

The process becomes quite similar to “Round”. Then the code for for AES encryption and decryption can be shared.

Problem 7

3DES remains in use for legacy systems where upgrading to AES is not feasible. In places where there are hardware constraints, e.g. older embedded systems with limited processing power or memory, 3DES is preferred due to its lower computational resource need.

There are various advantage of AES over 3DES. Firstly, AES is more secure since it has larger key size and more sophisticated encryption process. Secondly, AES is generally faster and more efficient. Lastly, AES offers more flexibility in key sizes(128, 192 and 256 bits), allowing even stronger encryption.

Problem 8

1. Establish a new key: Generate a new encryption key to replace the previous one. The new key must meet the latest cryptographic standards and security requirements.
2. Distribute the new key: The new key must be securely distributed to all the entities that require access to the encrypted sensitive data. This may involve encryption, digital signatures, or other secure mechanisms to ensure the integrity and confidentiality of the key distribution process.
3. Confirm the use of the new key: All the relevant systems and applications must be updated to use the new encryption key. This may involve upgrading systems, updating configurations, and ensuring that the new key is properly integrated into the organization's data protection processes.
4. Revoke the old key: Once the organization has confirmed that the new key is fully in use, the old key should be immediately revoked. This typically involves updating the key management system and publishing the revocation information, such as through a certificate revocation list.

Problem 9

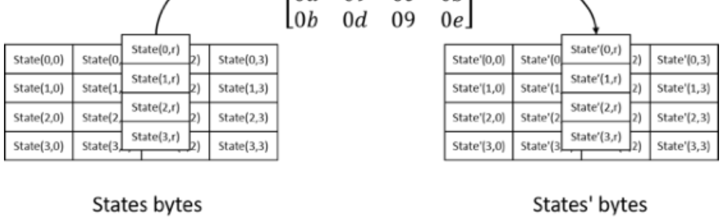
a) Firstly, Alice is not changing the modulus n. The new key pairs are not truly new, resulting in the lack of diversity over time. Secondly, the choice of e starting from 3 is too small, making the encryption extremely vulnerable. Lastly, the pattern of e values is highly predictable, making Alice an easy target for the attackers.

b) If the initial choice for p and q are not large enough, some attackers might be able to factor n and break the encryption. Moreover, the prime factor update exists a pattern and might lead to vulnerability. It will be easy for attackers to anticipate future results.

Problem 10

a) Each byte in the current state matrix and the given inverse mix-column matrix is treated as a polynomial in $GF(2^8)$. For example, 0E, is 0000 1110 in binary and is $x^3 + x^2 + x$ in $GF(2^8)$. The state matrix is a column matrix and is multiplied with each row of given inverse mix-column matrix byte by byte. The multiplication is performed modulo the irreducible polynomial, so the result must still be in $GF(2^8)$. Then do XOR to each result to get the new state. There are various differences from standard matrix multiplication. It uses XOR to combine the results instead of plus, and do modulo the irreducible polynomial after the XOR.

$$[S'] = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} * [S]$$



b) multiplication by 9: $\text{mul9}(x) = \text{mul2}(\text{mul2}(\text{mul2}(x))) \oplus x$

multiplication by 11: $\text{mul11}(x) = \text{mul2}(\text{mul2}(\text{mul2}(x))) \oplus \text{mul2}(x) \oplus x$

multiplication by 13: $\text{mul13}(x) = \text{mul2}(\text{mul2}(\text{mul2}(x))) \oplus \text{mul2}(\text{mul2}(x)) \oplus x$

multiplication by 14: $\text{mul14}(x) = \text{mul2}(\text{mul2}(\text{mul2}(x))) \oplus \text{mul2}(\text{mul2}(x)) \oplus \text{mul2}(x)$

c) LUTs can store the values of results of 256 possible inputs multiplied by 09, 0B, 0D and 0E. Then the process of multiplication becomes a simple table lookup and XORs. The advantages are that the multiplication process can be faster, more consistent and correct. The potential drawbacks are the requirement of additional memory of storing LUTs, and the increased vulnerability to side-channel attacks and more. The access pattern to the LUTs can leak information about the encryption key.

Problem 11

According to the diagram of PBC:

$c_0 = E_k(m_0) \oplus IV$

$c_1 = E_k(m_1) \oplus m_0$

$c_2 = E_k(m_2) \oplus m_1$

Since $m_1 = m_2 = X$ (known), we get

$c_2 = E_k(X) \oplus X$

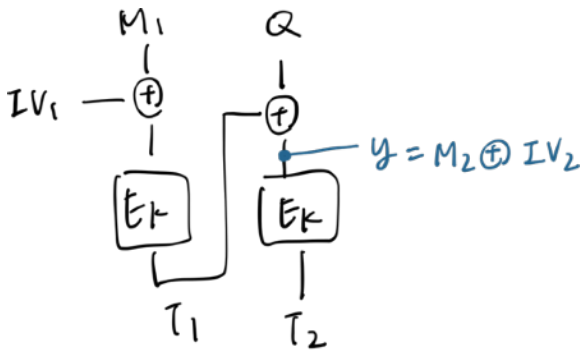
then we know the encryption $E_k(X)$, since c_2 and X are known

Solve $c_1 = E_k(X) \oplus m_0$ and $c_0 = E_k(m_0) \oplus IV$

to get the value of m_0.

Problem 12

Start with two messages M_1 and M_2, with respective outputs (IV_1, T_1) and (IV_2, T_2). Construct a message $M_3 = M_1 || Q$, where $Q = M_2 \oplus IV_2 \oplus T_1$. Then M_3 has output (IV_1, T_2).



$Q \oplus T_1 = M_2 \oplus IV_2$

$Q = M_2 \oplus IV_2 \oplus T_1$

Send M_3 with its MAC (IV_1, T_2), then the receiver will get the same tag T_2 as M_2. Then an existential forgery is produced.