CE Midterm

學號: 113550021 姓名: 陳孟楷

建議使用HackMD瀏覽，傳送門

Problem 1

Question a.

$f_1 + g_1 = x^3 + (x^2 + x^2) + (1 + 1) = \underline{x^3}$
In $\mathbb{F}_2$, addtion and subtraction are the same.
$f_1 - g_1 = f_1 + g_1 = \underline{x^3}$

$f_1 \cdot g_1 = (x^2 + 1)(x^3 + x^2 + 1) = x^5 + x^4 + x^3 + 1$
By Remainder Theorem, if we want to get $f_1 \cdot g_1 \; mod \; P(x)$, we can let $P(x) = 0$ to get the remainder.
$x^4 + x + 1 = 0 \Rightarrow x^4 \equiv x + 1$
$x^5 = x \cdot x^4 = x(x + 1) = x^2 + x$
$f_1 \cdot g_1 \; mod \; P(x) = x^2 + x + x + 1 + x^3 + 1 = \underline{x^3 + x^2}$

$f_2 + g_2 = x^2 + 1 + x + 1 = \underline{x^2 + x}$
$f_2 - g_2 = f_2 + g_2 = \underline{x^2 + x}$

$f_2 \cdot g_2 = (x^2 + 1)(x + 1) = x^3 + x^2 + x + 1$
$f_2 \cdot g_2 \; modP(x) \; = (x^2 + 1)(x + 1) = \underline{x^3 + x^2 + x + 1}$

Question b-a.

We test all polynomials of degree 1 and 2 over $\mathbb{F}_2$ to see if they divide $P(x)$.
1st-degree: We have x and $x + 1$.
2nd-degree: We have x^2, $x^2 + 1$, $x^2 + x$, and $x^2 + x + 1$.
However, all these polynomials don't divide $P(x)$.
Therefore, $P(x)$ is irreducible.

Question b-b.

There are $2^4 - 1 = 15$ nonzero elements. The order of x must divide 15. If we compute $x^k \; mod \; P(x)$ and the smallest k such that $x^k = 1$ is 15.
So, the answer is $\underline{15}$.

Question b-c.

Yes, because x has multiplicative order 15 in $\mathbb{F}_{2^4}$, which is the maximum possible.
So, $P(x)$ is primitive.

Problem 2

Question a.

We test all polynomials of degree 1 and 2 over $\mathbb{F}_2$ to see if they divide $p(x)$.
1st-degree: We have x and $x+1$.
2nd-degree: We have x^2, x^2+1, x^2+x, and x^2+x+1.
However, all these polynomials don't divide $p(x)$.
Therefore, $p(x)$ is irreducible.

Question b-a.

Use the same way as Problem a.-Question a., we have $x^4 \equiv x+1$, $x^5 \equiv x^2+x, \dots$
If we can simplify x^n to 1 while $mod\, p(x)$, $p(x)$ can divide x^n-1.
Otherwise, it is not divisible.

$n=1$, $x \bmod P(x) \equiv x$
$n=2$, $x^2 \bmod P(x) \equiv x^2$
$n=3$, $x^3 \bmod P(x) \equiv x^3$
$n=4$, $x^4 \bmod P(x) \equiv x+1$
$n=5$, $x^5 \bmod P(x) \equiv x^2+x$
$n=6$, $x^6 \bmod P(x) \equiv x^3+x^2$
$n=7$, $x^7 \bmod P(x) \equiv x^4+x^3 \equiv (x+1)+x^3 \equiv x^3+x+1$
$n=8$, $x^8 \bmod P(x) \equiv (x+1)^2 \equiv x^2+1$
$n=9$, $x^9 \bmod P(x) \equiv x\cdot(x+1)^2 \equiv x^3+x$
$n=10$, $x^{10} \bmod P(x) \equiv x^2\cdot(x+1)^2 \equiv x^4+x^2 \equiv x^2+x+1$
$n=11$, $x^{11} \bmod P(x) \equiv x^3\cdot(x+1)^2 \equiv x^3+x^2+x$
$n=12$, $x^{12} \bmod P(x) \equiv (x+1)^3 \equiv x^3+3x^2+3x+1 \equiv x^3+x^2+x+1$
$n=13$, $x^{13} \bmod P(x) \equiv x\cdot(x+1)^3 \equiv x^4+x^3+x^2+x \equiv x^3+x^2+1$
$n=14$, $x^{14} \bmod P(x) \equiv x^2\cdot(x+1)^3 \equiv x^4+x^3+x \equiv x^3+1$

Question b-b.

$n=15$, $x^{15} \bmod P(x) \equiv x^3\cdot(x+1)^3 \equiv x^4+x \equiv 1$
$x^{15} \equiv 1 \bmod P(x) \Rightarrow x^{15}-1 \equiv 0 \mod p(x)$
So, $p(x)$ divides $x^{15}-1$.

Question c.

Relation

Primitive polynomials 在 $GF(2)$ 上是 irreducible 的，可以產生所有 2^n-1 種非零狀態，稱為

maximal-length LSFR sequence。

Why generate maximal-length sequence

因為 $x^4 + x + 1$ 是 irreducible 且 primitive，可以產生 15 個非零狀態，其所產生的序列為 000100110101111100，而 s 正為其中一部分。

Question d.

Irreducible polynomials

$x^4 + x + 1$

$x^4 + x^3 + 1$

$x^4 + x^3 + x^2 + x + 1$

Primitive polynomials

$x^4 + x + 1$

$x^4 + x^3 + 1$

Method

Irreducible polynomials 中，multiplicative order 為 $2^n - 1$ 的就是 primitive polynomials.

Problem 3

Question a.

1. Step 1

- Berlekamp–Massey 演算法需要 $2n$ 長度的序列來求解長度為 n 的 LFSR。
- 因此，使用 s 最多只能求得長度為 4 的 LFSR。

2. Step 2

- 我們需要求解一組遞迴關係：$s = (0, 0, 1, 1, 0, 1, 0, 1, 1)$
- 利用已知序列帶入計算，得到以下四個方程：
    1. $0 = 0C_3 + 1C_2 + 1C_1 + 0C_0 \ mod \ 2$
    2. $0 = 1C_3 + 1C_2 + 0C_1 + 1C_0 \ mod \ 2$
    3. $1 = 1C_3 + 0C_2 + 1C_1 + 0C_0 \ mod \ 2$
    4. $1 = 0C_3 + 1C_2 + 0C_1 + 1C_0 \ mod \ 2$

3. Step 3

- 解這組方程組，得到: $C_3 = 1, C_0 = 1, C_2 = 0, C_1 = 0$
- 因此，LFSR 的特徵多項式為：$C_{BM}(x) = x^4 + x + 1$

Question b.

1. Step 1

- 利用序列 s 建立多項式：$S(x) = x^6 + x^5 + x^3 + x + 1$
- 假設 LFSR 的長度為 4，則特徵多項式為：

2. Step 2

- 利用 EEA 計算 GCD: $GCD(x^8, S(x)) = C_{BM}(x) = x^4 + x + 1$

Question c.

Compare

BMA 和 EEA 的結果相同。

Explain

EEA 找到的 GCD 實際上是最小多項式，因為它是序列多項式的最大公因數（最低次多項式）。
BMA 則是從序列本身動態構造出這個最小多項式，並確保它符合序列的遞迴關係。
兩者都針對序列進行運算，並且都保證能找到唯一的最小多項式。

Problem 4

Question a.

Define M

M 是是通過循環旋轉二進位向量 11110000 來構造的 8x8 矩陣：

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Invertible Proof

- 將 matrix 做 Gaussian elimination 後，可以找到 all-zero 的 row，故此 matrix 為 invertible。

Reference: Generation of AES-Like S-Boxes by Replacing Affine Matrix

Question b.

1. 反向運算

- 在有限域 $GF(2^8)$ 上計算 x_{-1}，使用多項式 $p_2(x) = x^8 + x^5 + x^3 + x + 1$。
- 如果輸入 $x = 0$，則輸出也為 0。
- 否則，計算 $x^{254}\ mod\ p_2(x)$，這是有限域中反向的快速計算方法。

2. 仿射轉換:

- 使用循環旋轉構造的 8x8 矩陣:

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- 常數: $c = 0x63$。
- 計算: $S(x) = M \times x^{-1} \oplus c$

```
def s_box(x):
    if x == 0:
        inv_x = 0  # Special case: 0 has no inverse
    else:
        inv_x = pow(x, 254, 0x11B)  # GF(2^8) inverse using modulus of p2(x)

    m = 0b11110000
    s = 0
    for i in range(8):
        s ^= ((inv_x >> i) & 1) * ((m >> i) & 1)
    s ^= 0x63
    return s
for i in range(256):
    print(f"S({i}) = {s_box(i):02X}")
```

Question d.

1. 數學合理性：

- 循環旋轉 11110000 會生成一個有系統且可逆的 8x8 矩陣。
- 這種構造方式保證了每一位元組都受到多個位元影響，達到 位元擴散（Bit Diffusion）。
- 在 AES 這種代數強度高的 S-Box 設計中，擴散是必不可少的。

2. 安全性分析：

- 若旋轉後的矩陣滿足可逆性，且擁有較高的非線性度，則它在加密中可以有效抵抗線性攻擊和差分攻擊。
- 這種構造與傳統 AES S-Box 的仿射層原理類似：利用線性變換加強 S-Box 的安全性。

Reference: The Design of Rijndael: Advanced Encryption Standard (Aes)

Problem 5

Question a.

操作	標準實現	T-box 實現
SubBytes	為每個字節查找 S-box	通過 SB(x) 融入 T-box 表項
ShiftRows	循環移位行	通過查找的字節選擇處理
MixColumns	在 GF(2^8) 中進行矩陣乘法	預計算於 T-box 表項中

T-box 實現通過預計算包含 SubBytes 和 MixColumns 的表，並通過輸入字節的索引方式融入 ShiftRows，從而合併 SubBytes、ShiftRows 和 MixColumns。此方法顯著提升性能，降低計算複雜度和延遲，是高效 AES 實現的首選方法。

Question b.

Constant-time 的設計原則:

- 固定大小查表(256個條目)
- 固定時間查表(constant-time lookup)
- 使用位元遮罩(bitwise AND)進行選擇

固定大小查表(256個條目):

```
def precompute_mixcolumns_lut():
    lut = np.zeros((256, 4), dtype=np.uint8)
    for i in range(256):
        lut[i] = mixcolumns_single_byte(i)
    return lut
```

固定時間查表(constant-time lookup)

```
def constant_time_lookup(lut, byte):
    result = np.zeros(4, dtype=np.uint8)
    for i in range(256):
        mask = -(i == byte) & 0xFF
        result ^= lut[i] * mask
    return result
```

使用位元遮罩(bitwise AND)進行選擇:

```
mask = -(i == byte) & 0xFF
```

Question c.

性能	固定時間 T-box 實作	位切片（Bit-Sliced）實作

Memory Footprint	4 KB	~300 bytes (平行處理情況下)
Latency	~30 cycles/block	~50–80 cycles/block
Rough Cycle Counts	25~35 cycles/block	50~80 cycles/block

Problem 6

Operations Ordering

The order of InvSubBytes and InvShiftRows is indifferent:
在AES解密中，因為 `InvSubBytes` 和 `InvShiftRows` 這兩個operations在每個字節中是獨立進行的。

The order of AddRoundKey and InvMixColumns can be inverted if the round key is adapted accordingly:
因為 `MixColumns` 是一個線性操作，且可以將金鑰視為乘數進行調整。所以，AddRoundKey 和 InvMixColumns 的順序可以互換。

Gate-count savings

- AES： 若共用程式碼，可節省 20% 至 30% 的指令數。
- ASIC： 可節省 10% 至 15% 的晶片區域，主要在 S-box 及逆 S-box 共享。

Problem 7

Question a.

- Irreducible: 在 $\mathbb{F}_2$ 上定義，無法被分解成低次多項式的多項式。
- Primitive: 在 $\mathbb{F}_2$ 上定義的 irreducible polynomial，且它的根生成有限域 $\mathbb{F}_{2^n}^*$ 中的所有非零元素。
- Relationship: 所有的 primitive polynomial 都是 irreducible，但並不是所有的 irreducible polynomial 都 primitive。

Question b-a.

We test all polynomials of degree 1 and 2 over $\mathbb{F}_2$ to see if they divide $p(x)$.
1st-degree: We have x and $x+1$.
2nd-degree: We have x^2, x^2+1, x^2+x, and x^2+x+1.
However, all these polynomials don't divide $p(x)$.
Therefore, $p(x)$ is irreducible.

Question b-b.

Yes, 一個irreducible polynomial 如果是primitive，它的根 α 應該要是 $\mathbb{F}_{2^4}$ 的本原元素。
這意味著 α 的階數應該是$2^4 - 1 = 15$，也就是說:
$\alpha^{15} = 1$ 且沒有小於 15 的整數 k 使得 $\alpha^k = 1$。
因為α是$p(x)$的一個根，$\alpha^4 = \alpha^3 + 1$

proof:

1. $\alpha^1 \equiv \alpha$
2. $\alpha^2 \equiv \alpha^2$
3. $\alpha^3 \equiv \alpha^3$
4. $\alpha^4 \equiv \alpha^3 + 1$
5. $\alpha^5 \equiv \alpha^3 + \alpha + 1$
6. $\alpha^6 \equiv \alpha^3 + \alpha^2 + \alpha$
7. $\alpha^7 \equiv \alpha^3 + \alpha^2 + \alpha + 1$
8. $\alpha^8 \equiv \alpha^3 + \alpha^2 + 1$
9. $\alpha^9 \equiv \alpha^3 + \alpha^2$
10. $\alpha^{10} \equiv \alpha^3 + \alpha + 1$
11. $\alpha^{11} \equiv \alpha^3 + 1$
12. $\alpha^{12} \equiv \alpha^3$
13. $\alpha^{13} \equiv \alpha^2$
14. $\alpha^{14} \equiv \alpha$
15. $\alpha^{15} \equiv 1$

Question c-a.

We test all polynomials of degree 1 and 2 over $\mathbb{F}_2$ to see if they divide $q(x)$.
1st-degree: We have x and $x + 1$.
2nd-degree: We have x^2, $x^2 + 1$, $x^2 + x$, and $x^2 + x + 1$.
However, all these polynomials don't divide $q(x)$.
Therefore, $q(x)$ is irreducible.

Question c-b.

Yes, 一個 irreducible polynomial 如果是 primitive，它的根 α 應該要是 $\mathbb{F}_{2^4}$ 的本原元素。
這意味著 α 的階數應該是$2^4 - 1 = 15$，也就是說:
$\alpha^{15} = 1$ 且沒有小於 15 的整數 k 使得 $\alpha^k = 1$。
因為α是$p(x)$的一個根，$\alpha^4 = \alpha + 1$

proof:

1. $\alpha^1 \equiv \alpha$
2. $\alpha^2 \equiv \alpha^2$

3. $\alpha^3 \equiv \alpha^3$
4. $\alpha^4 \equiv \alpha + 1$
5. $\alpha^5 \equiv \alpha^2 + \alpha$
6. $\alpha^6 \equiv \alpha^3 + \alpha^2$
7. $\alpha^7 \equiv \alpha^3 + \alpha + 1$
8. $\alpha^8 \equiv \alpha^3 + \alpha^2 + \alpha$
9. $\alpha^9 \equiv \alpha^3 + \alpha^2 + \alpha + 1$
10. $\alpha^{10} \equiv \alpha^3 + \alpha^2 + 1$
11. $\alpha^{11} \equiv \alpha^3 + \alpha$
12. $\alpha^{12} \equiv \alpha^3 + 1$
13. $\alpha^{13} \equiv \alpha^3$
14. $\alpha^{14} \equiv \alpha^2$
15. $\alpha^{15} \equiv 1$

Question d.

Connection:
當一個LFSR基於primitive polynomial時，它可以生成maximal-length sequence，也稱作m-sequence。
Period:
如果一個LFSR是基於階數為 n 的primitive polynomial，那麼period就是 $2^n - 1$
Difference with irreducible polynomial but not primitive:
週期會小於 $2^n - 1$，因為這樣的多項式雖然無法分解，但它的根的階數不足 $2^n - 1$。

Problem 8

Question a-a.

一個加密系統具有完美保密性，當加密後的密文不會洩露任何關於明文的資訊，即使攻擊者知道加密演算法。數學表示為： $P(m|c) = P(m)$

Question a-b.

為了達成完美保密性，每個明文必須能夠以相同機率對應到每個密文。
這表示每個密文必須唯一對應於一個明文，這種對應取決於金鑰。
因此，金鑰數量至少要與明文數量相等：$|K| \geq |M|$

Question a-c.

Perfect Secrecy
One-Time Pad (OTP) 是唯一能達到「完全保密性 (Perfect Secrecy)」的對稱加密方法。它的安全

性來源於以下特性：

1. 隨機性： OTP 的密鑰必須是隨機產生且不重複，這意味著每一個密鑰位元（bit）完全不可預測。
2. 等長密鑰： OTP 的密鑰長度必須與明文相同。這意味著每一位明文會被一位隨機密鑰位元進行異或（XOR）運算，生成相應的密文位元：$C_i = P_i \oplus K_i$, C_i 是密文位元, P_i 是明文位元, K_i 是隨機密鑰位元。
3. 理論上的不可破解： 由於每個密文位元是隨機的，攻擊者無法從密文中推斷出任何關於明文的資訊。即使攻擊者截獲密文，沒有密鑰的情況下也無法得知明文內容。

Impractical in real-world applications

1. 密鑰分配 (Key Distribution)：

- 每次加密都需要一條長度等於明文的隨機密鑰。
- 這些密鑰必須安全地傳輸給接收者。
- 如果密鑰被攔截或洩漏，攻擊者就能解密所有通信。

2. 密鑰儲存 (Key Storage)：

- 每個通信雙方必須安全儲存這些密鑰，並且確保它們不會被洩漏或重複使用。
- 隨著通信次數的增加，密鑰管理的需求會呈指數增長。

3. 隨機性要求：

- 生成真正隨機的密鑰是極具挑戰性的，尤其是在高效能的系統中。
- 大多數電腦隨機數產生器 (PRNG) 並非真正隨機，而是偽隨機 (Pseudorandom)，這會降低安全性。

4. 單次使用 (One-Time Use)：

- 一旦密鑰被重複使用（即使是部分重複），OTP 的安全性將被完全破壞。這違反了「One-Time」的原則。

Question b-a.

$$X_1 = (75 \cdot 12345 + 74) \; mod \; 65536 = 8445$$
$$X_2 = (75 \cdot 8445 + 74) \; mod \; 65536 = 43625$$
$$X_3 = (75 \cdot 43625 + 74) \; mod \; 65536 = 60685$$
$$X_4 = (75 \cdot 60685 + 74) \; mod \; 65536 = 29465$$
$$X_5 = (75 \cdot 29465 + 74) \; mod \; 65536 = 47261$$

Question b-b.

1. 周期性與可預測性：

- LCG 是一種確定性隨機數生成器，其輸出序列最終會循環。
- 由於 LCG 的參數（a, c, m）是固定的，攻擊者可以通過少量輸出推算出整個序列。

2. 線性特性：

- 由於 LCG 是線性的（即下一個輸出是上一個輸出的線性組合），攻擊者可以使用線性代數推算出內部參數（a, c, m）。
- 只需要觀察幾個輸出，攻擊者就可以建立聯立方程組來破解 LCG。

Question c-a.

- irreducible: $p(x)$ 是 irreducible，因為它無法在 $\mathbb{F}_2$ 上進行因式分解
- primitive: $p(x)$ 是 primitive，因為能夠在 $GF(2^8)$ 中生成255個唯一的非零元素，形成一個完整的循環序列。

Question c-b.

當攻擊者觀察到 LFSR 產生的 20 個連續位元時，他可以利用 Berlekamp-Massey 演算法來重建整個 LFSR。這是因為 LFSR 的長度為 8 位元，而 Berlekamp-Massey 演算法可以在 2 倍於 LFSR 長度的位元內（即 16 位元）重建 LFSR 連接多項式。

Question c-c.

- 使用多個獨立的 LFSR 來產生序列: 雖然 LFSR 本質上是線性的，但多個 LFSR 的非線性組合極大地增加了序列的複雜度
- 將這些 LFSR 的輸出通過一個非線性組合函數，以產生最終的序列輸出: Berlekamp-Massey 演算法只能針對單一線性序列進行攻擊，無法處理這種多重非線性結構。

Problem 9

Question a.

$f(x) = ax^2 + bx + c$，判別式 $\triangle = b^2 - 4ac$
當 $\triangle > 0$ 時，$f(x)$ 有兩個解。
當 $\triangle = 0$ 時，$f(x)$ 有一個解。
當 $\triangle < 0$ 時，$f(x)$ 有零個解（無解）。

Question b.

因為 $a^5 = b^3 = e$，所以 n 必須是這兩個數的最小公倍數： $n = lcm(3, 5) = 15$
所以 ab 的 order 是 15。

Question c-a.

可以計算 $y^{(p-1)/2} \, mod \, p$ 的值

- 若結果為 1，則 y is quadratic residue
- 若結果為 -1 或 $p-1$，則 y is not quadratic residue

Question c-b.

Bob can send a non-quadratic residue, however, he can't cheat Alice since Alice can use the way in Question c-a. to check whether it's a quadratic residue.

Question c-c.

1. 防禦 Man-in-the-middle

- 在安全密鑰交換協定（如 Diffie-Hellman）中，二次剩餘可用來確認交換的值是合法的。
- 如果攻擊者 (Eve) 企圖進行中間人攻擊，她必須能夠生成有效的二次剩餘，這在沒有私鑰的情況下極為困難。
- 這種驗證機制可確保雙方通訊的正確性。

2. 安全加密：Goldwasser-Micali 加密系統

- Goldwasser-Micali 是一種基於二次剩餘的加密系統：
  - 若訊息位元 $m=0$, 則密文是二次剩餘。
  - 若 $m=1$, 則密文是非二次剩餘。
- 因為在不知密鑰的情況下，無法區分二次剩餘和非二次剩餘，因此該加密系統在數學上具有強大的安全性。

Question d.

- small prime e : small prime e can accelerate the encryption calcalation
- large, random d : Since d is quite large, it's difficult to perform 質因數分解 on it.

Therefore, choose small prime e and large, random d can promise the safety.

Problem 10

Key Revocation Process

- 立即禁用密鑰
- 通知相關人員
- 審計與記錄
- 進入後操作階段

New Key Deployment

- 生成新金鑰

- 密鑰分發
- 加密演算法更新

Incident Response

時間點	行動
0-1 小時內	偵測並確認金鑰洩漏，禁止舊金鑰使用
1-2 小時內	通知管理層與受影響人員
2-4 小時內	生成並分發新金鑰
4-8 小時內	調查洩漏來源，保護相關證據
8-24 小時內	更新金鑰管理策略，審計事件

Reference : NIST Special Publication 800-57 Part 1

Problem 11

Question a.

以兩組公鑰$(n, e_1), (n, e_2)$ 為例，
attacker 可以透過歐基里德算法，找到$a \cdot e_1 + b \cdot e_2 = 1$, 則 $m = c_1^a \cdot c_2^b \, mod \, n$，可以恢復明文。

Question b.

Let $p_1 = p + \delta_1$ and $q_1 = q + \delta_1$,
then $n_1 = p \times q$
$n_2 = p_1 \times q_1 = (p + \delta_1)(q + \delta_2) = pq + p\delta_2 + \delta_1 q + \delta_1\delta_2$
$n_2 = n_1 + p\delta_2 + \delta_1 q + \delta_1\delta_2$

Since $\delta_1\delta_2$ is quite small compared to $p\delta_2$ and $\delta_1 q$
$n_2 - n_1 \approx p\delta_2 + \delta_1 q$

We define a lattice basis B:

$$\begin{pmatrix} \delta_2 & \Delta \\ -\delta_1 & 0 \end{pmatrix}$$

Our objective is to find the shortest vector in this lattice, which will approximate (p, q).
Using LLL algorithm, we can recover δ_1 and δ_2, leading to the recovery of p and q.

Problem 12

Question a.

- Brief Explaination and Difference

Matrix addition in $GF(2^8)$，就是按位 XOR。

Matrix multiplication in $GF(2^8)$ is similar normal matrix multiplication.

Multiply the elements in the row of the first matrix and those in the column of the second matrix, respectively.

However, the difference is that changing the "plus sign" into "XOR sign".

For example,

Matrix A:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}$$

Matrix B:

$$\begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{bmatrix}$$

In $GF(2^8)$,
$S'_0 = (0E \times S_0) \oplus (0B \times S_1) \oplus (0D \times S_2) \oplus (09 \times S_3)$
$S'_1 = (09 \times S_0) \oplus (0E \times S_1) \oplus (0B \times S_2) \oplus (0D \times S_3)$
$S'_0 = (0D \times S_0) \oplus (09 \times S_1) \oplus (0E \times S_2) \oplus (0B \times S_3)$
$S'_0 = (0B \times S_0) \oplus (0D \times S_1) \oplus (09 \times S_2) \oplus (0E \times S_3)$

In standard matrix multiplication,
$S'_0 = (0E \times S_0) + (0B \times S_1) + (0D \times S_2) + (09 \times S_3)$
$S'_1 = (09 \times S_0) + (0E \times S_1) + (0B \times S_2) + (0D \times S_3)$
$S'_2 = (0D \times S_0) + (09 \times S_1) + (0E \times S_2) + (0B \times S_3)$
$S'_3 = (0B \times S_0) + (0D \times S_1) + (09 \times S_2) + (0E \times S_3)$

Question b.

- $S \times 9 = (S \times 8) \oplus S = (((S \times 2) \times 2) \times 2) \oplus S$
- $S \times 11 = (S \times 8) \oplus (S \times 2) \oplus S = (((S \times 2) \times 2) \times 2) \oplus (S \times 2) \oplus S$
- $S \times 13 = (S \times 8) \oplus (S \times 4) \oplus S = (((S \times 2) \times 2) \times 2) \oplus ((S \times 2) \times 2) \oplus S$

- $S \times 14 = (S \times 8) \oplus (S \times 4) \oplus (S \times 2) = (((S \times 2) \times 2) \times 2) \oplus ((S \times 2) \times 2) \oplus (S \times 2)$

Question c.

- Advantage
  - **高速**： 使用查找表來執行乘法和 XOR 可以顯著加快運算速度，因為表格查找是 $O(1)$
  - **簡化設計**： 可以預先計算所有可能的 $GF(2^8)$ 乘法結果，並儲存在表格中。
  - **硬體資源優化**： 在硬體中，LUT 只需定義一次，無需動態計算。
- Drawback
  - **記憶體需求**： LUT 需要儲存所有可能的結果，這會佔用相當的記憶體空間（如 256×256 表格）。
  - **側信道風險**： 查找表可能暴露於側信道攻擊（如快取計時攻擊），如果表格訪問的時間可被觀察。

Question d.

$9, 11, 13, 14$ 最多 shift 3 次 $(\times 2, \times 4, \times 8)$，故 logical depth 為 3。
area:
$9a = 8a \oplus a \to 1$
$11a = 8a \oplus 2a \oplus a \to 2$
$13a = 8a \oplus 4a \oplus a \to 2$
$14a = 8a \oplus 4a \oplus 2a \to 2$
總area: 3 multipliers + 7 XOR gates

Problem 13

Attack Demonstration

Let E be the encryption function.
$C_0 = E(IV \oplus m_0)$
$C_1 = E(C_0 \oplus m_1) = E(C_0 \oplus x)$
$C_2 = E(C_1 \oplus m_2) = E(C_1 \oplus x)$

$IV \oplus m_0 = C_0 \oplus x \oplus C_1$
$m_0 = (C_0 \oplus x \oplus C_1) \oplus IV$

Distinguishing Advantage

$\varepsilon(n) = 1 - \frac{1}{2^n}$
當區塊長度 n 增加時，區分優勢趨近於 1，表示攻擊成功的概率極高。

Problem 14

Existential Forgery Attack

- Construct M_3
  - M_3 是一個新訊息，其內容接近於 $M_1||M_2$ (M_1 和 M_2 連接起來)。
  - 修改 M_3 的最後一個區塊，使其與 M_2 的最後一個區塊不同。
- Implementation
  - Let M_3 be $M_1||M_2^*$, M_2^* 是 M_2 的一個變異版本，且僅改變最後一個區塊 B : $M_2^* = M_2[0], M_2[1], \ldots, M_2[n-1], B$
  - 此時，透過 CBC-MAC 計算: $T_3 = MAC(IV_1, M_3)$
  - 由於 CBC-MAC 僅依賴最後一個區塊進行輸出，且我們已知 T_2，我們可以透過選擇 B 使得: $T_3 = T_2$
  - 這代表我們已經成功的偽造一訊息 M_3 並產生 $MAC(IV_1, T_2)$

Variable-length Generalization

- Construct M_3 為可變長度: $M_3 = M_1||(M_2^* \oplus T_1)$
  - M_2^* 是 M_2 的第一個區塊 B_0
  - 將 B_0 進行 XOR 運算: $M_2^*[0] = M_2[0] \oplus T_1$
  - M_3 可以是任意長度的訊息
- Implementation
  - 前半部分 M_1 計算至 MAC 值 T_1
  - T_1 被當作下一區塊的輸入，進行XOR: $T_1 \oplus (M_2[0] \oplus T_1) = M_2[0]$
  - 後續 M_2 的計算如同原本 M_2 訊息: $MAC(IV_1, M_3) = T_2$
- Proof
  - 設 M_1 為 $B_1, B_2, \ldots, B_n$, MAC 為 T_1: $T_1 = MAC(IV_1, M_1)$
  - 設 M_1 為 $B_3, B_4, \ldots, B_m$, MAC 為 T_2: $T_2 = MAC(IV_2, M_2)$
  - 攻擊者構造 M_3 為: $M_3 = B_1, B_2, \ldots, B_n, (B_3 \oplus T_1), B_4, .., B_m$
  - CBC-MAC 對 M_3 的計算:
    - $T_1 = E_K(IV_1 \oplus B_1) \rightarrow E_K(T_1 \oplus B_2) \rightarrow \ldots \rightarrow E_K(T_1)$
    - $E_K(T_1 \oplus (B_3 \oplus T_1)) = E_K(B_3)$
    - $E_K(B_3) \rightarrow \ldots \rightarrow T_2$
  - 攻擊者在選擇訊息查詢下，成功偽造 M_3 並獲得 T_2。

Problem 15

Approach to Broadcast Attack

當多個接收者使用相同的公開指數 $e = 3$ 且每個人都有不同的模數 n_A, n_B, n_C時，如果 David 將相同的訊息 m 發送給這三個人，將導致安全漏洞。

Eve 可以攔截這三個加密的訊息 y_A, y_B, y_C，並利用這些訊息計算出明文 m 而無需知道這三個使用者的私鑰。

CRT Application and Proof

By Chinese Remainder Theorem
$y \equiv y_A \ (mod \, n_A)$
$y \equiv y_B \ (mod \, n_B)$
$y \equiv y_C \ (mod \, n_C)$

這意味著可以計算一個 y 使得:
$y = m^3 (mod \, N)$
$N = n_A \times n_B \times n_C$

Plaintext Extraction Procedure

We have $y = m^3 \ mod \, N$，可以通過計算三次方根來恢復 m : $m = y^{1/3} \ (mod \, N)$

Problem 16

PFS Protocol Specification

使用 Diffie-Hellman (DH) 密鑰交換來實現完全前向安全性:

1. 初始化：Alice 和 Bob 彼此交換各自的公開金鑰

- Alice 的公開金鑰: $g^a \ mod \, p$，期中 g 是基數，p 是大質數，a 是 Alice 的金鑰。
- Bob 的公開金鑰: $g^b \ mod \, p$，其中 b 是 Bob 的私鑰。

2. DH 金鑰交換

- Alice 使用 Bob 的公開金鑰計算會話金鑰: $K_{AB} = (g^b)^a \ mod \, p$
- Bob 使用 Alice 的公開金鑰計算相同的會話金鑰: $K_{AB} = (g^a)^b \ mod \, p$
- 雖然 Alice 和 Bob 彼此交換了公開金鑰，但只有他們知道各自的私鑰 (a, b)，因此只有他們能計算出此會話金鑰 K_{AB}。

3. 加密通訊

- Alice 和 Bob 使用 K_{AB} 作為對稱加密金鑰來加密彼此之間的通訊。
- 每次進行通訊時，雙方都進行新的 DH 金鑰交換，生成新的 K_{AB}，即使雙方的長期密鑰保持不變。

Forward Secrecy Proof

Eve 雖然獲得了 a 和 b，但這些只是 Alice 和 Bob 用於產生 DH 會話金鑰的私鑰。

由於每次通訊 Alice 和 Bob 都進行新的 DH 金鑰交換，因此每次的會話金鑰 K_{AB} 都是獨立的。

過去的會話金鑰 K_{AB} 是臨時的，無法從 Alice 和 Bob 的長期私鑰中還原。

Eve 只能看到加密的通訊內容和公開金鑰，無法計算每次通訊的 K_{AB} 值。

Real-world Implementation Evaluation

在現實中，PFS 被廣泛應用於 TLS（傳輸層安全性）協定中，尤其是在 TLS 1.3 中的 ECDHE（Elliptic Curve Diffie-Hellman Ephemeral）。

- TLS 1.3 的 PFS：
  透過每次連線建立時的臨時 ECDH 參數，實現每次連線都擁有唯一的會話金鑰。
  即使伺服器的私鑰被洩露，攻擊者無法回溯解密先前的通訊。
- 好處：
  提升通訊安全性，保護歷史訊息。
  即使伺服器密鑰被洩露，也無法影響過去的連線安全。
- 範例應用：
  當你使用 HTTPS 連接到銀行網站或電子郵件伺服器時，這些網站會使用 PFS 來確保你過去的通訊不會因伺服器私鑰洩露而被解密。

Extra-Credit 1

Perfect Secrecy Proof

1. 由於 p 是質數，對於每個可能的密文 c，存在為唯一的 k 滿足: $m = c \times k^{-1} \mod p$，其中 k^{-1} 是 k 在模 p 下的乘法逆元。
2. 由於 k 是隨機選擇且均勻分布於 $\{1, 2, \ldots, p-1\}$，這意味著每個可能的 m 都是等機率的，因為每個 k 對應一個唯一的 m。
3. 因此，給定c，所有 m 都是等可能的，這正是完美保密性的定義。

LCG Cryptanalysis

- LCG 定義: $x_{i+1} = (ax_i + b) \mod p$，$a$ 是乘數，b 是偏移量，p 是模數，x_0 是初始狀態（種子）。

- 破解方法: 我們假設已觀察到 LCG 的輸出序列 $(x_0, x_1, x_2, .., x_n)$。

- 步驟:

1. 計算差分

- $d_1 = x_1 - x_0$，$d_1 = x_1 - x_0$，$d_2 = x_2 - x_1$，$d_n = x_n - x_{n-1}$。

2. 利用模數 p 簡化

- $d_{i+1} - d_i \equiv a(d_i - d_{i-1}) \, mod \, p$
- 這是一個線性方程組，攻擊者可以透過高斯消去法來計算 a 和 b。

3. 計算 p 的可能性

- 透過觀察最大公因數 $gcd(d_{i+1} - d_i)$，攻擊者可以推測 p 的可能值。

4. 計算 a, b 的確切值

- 將 a, b 代入方程，驗證是否滿足觀察到的序列。

5. 計算種子 x_0

- 利用已知序列 x_0 開始逐項驗證。

- LCG 為何不安全：
  - LCG 是線性的，這意味著它產生的序列可以被逆推。
  - 攻擊者只需要觀察幾個輸出即可建立線性方程組來推算參數。
  - 這是 LCG 在加密中不推薦的原因，因為它缺乏密碼學安全性。

Extra-Credit 2

First Non-linear Design

1. 設計原理

- 使用兩個或多個獨立的 LFSR，分別生成兩組或多組 LFSR 輸出序列:
  $S_1 = LFSR_1, \; S_2 = LFSR_2$
- 這兩組序列通過 XOR 運算組合成一個中間序列: $S_{mid} = S_1 \oplus S_2$
- 將這個中間序列輸入一個非線性 S-Box，生成最終輸出: $S_{out} = SBox(S_{mid})$
- 這種設計結合了多個 LFSR 的輸出，並通過非線性 S-Box 進一步增加了複雜度。

2. 安全性分析

- 由於使用了多個 LFSR，Berlekamp-Massey 無法直接破解整個輸出序列。
- S-Box 的非線性轉換進一步阻止了線性代數分析。
- 實際上，攻擊者需要猜測每個 LFSR 的狀態和 S-Box 的結構。

3. 硬體實現：

- 硬體門數主要來自 LFSR 和 S-Box。
- 假設每個 LFSR 需要 n 個觸發器和 XOR 閘，S-Box 為固定 256 字節查表，總門數約為: $2n + 256 \times 8 = 2n + 2048$
- 輸出速度等於 LFSR 的時鐘速率。

Second Design & Comparison

1. 設計原理：

- 使用一個 LFSR，但其反饋公式變為非線性：$x_{i+1} = f(x_i, x_{i-1}, \ldots, x_{i-k})$
- 期中 f 是一個非線性函數，如: $x_{i+1} = x_i \oplus (x_{i-1} \wedge x_{i-2}) \oplus SBox(x_{i-3})$
- 這意味著 LSFR 不再是線性的，而是依賴於前幾位的非線性組合。

2. 安全性分析

- Berlekamp-Massey 演算法針對線性序列，無法處理這種非線性回饋。
- 即使攻擊者觀察序列，也無法使用線性代數進行分析。
- 安全性主要取決於 S-Box 的強度和非線性公式的設計。

3. 硬體實現

- LFSR 的基本結構維持不變，僅在反饋路徑上增加非線性處理:
  - XOR 閘、AND 閘以及 S-Box 查表。
- 假設 S-Box 為 256 字節查表，總門數約為: $n + 2048 + (k \times 4)$
- 相比於第一種設計，這種方法能在相同的硬體規模下提高安全性。

Comparison

特性	第一種設計：多重 LFSR + S-Box	第二種設計：非線性反饋
安全性	高：結合多個 LFSR 和非線性 S-Box	極高：完全非線性反饋
硬體門數	2n + 2048	n + 2048 + (k × 4)
實作複雜度	中等：多個 LFSR + S-Box	高：非線性回饋
Berlekamp-Massey 攻擊	無效	完全無效
適用情境	高速流密碼生成	高安全性流密碼

Extra-Credit 3

Irreducible Proof

We test all polynomials of degree 1 and 2 over $\mathbb{F}_2$ to see if they divide $f_1(x)$ and $f_2(x)$.
1st-degree: We have x and $x+1$.
2nd-degree: We have x^2, x^2+1, x^2+x, and x^2+x+1.
However, all these polynomials don't divide $f_1(x)$ and $f_2(x)$.
Therefore, $f_1(x)$ and $f_2(x)$ irreducible.

Order Computations

There are $2^4-1=15$ nonzero elements. The order of x must divide 15. If we compute $x^k \; mod \; P(x)$ and the smallest k such that $x^k=1$ is 15.
So, the answer is $\underline{15}$.

Primitiveness analysis

一個 irreducible polynomial 如果是 primitive，它的根 α 應該要是 $\mathbb{F}_{2^4}$ 的本原元素。
這意味著 α 的階數應該是$2^4-1=15$，也就是說:
$\alpha^{15}=1$ 且沒有小於 15 的整數 k 使得 $\alpha^k=1$。
因為 α 是 $f_1(x)$ 的一個根，$\alpha^4=\alpha^3+1$
proof:

1. $\alpha^1 \equiv \alpha$
2. $\alpha^2 \equiv \alpha^2$
3. $\alpha^3 \equiv \alpha^3$
4. $\alpha^4 \equiv \alpha^3+1$
5. $\alpha^5 \equiv \alpha^3+\alpha+1$
6. $\alpha^6 \equiv \alpha^3+\alpha^2+\alpha$
7. $\alpha^7 \equiv \alpha^3+\alpha^2+\alpha+1$
8. $\alpha^8 \equiv \alpha^3+\alpha^2+1$
9. $\alpha^9 \equiv \alpha^3+\alpha^2$
10. $\alpha^{10} \equiv \alpha^3+\alpha+1$
11. $\alpha^{11} \equiv \alpha^3+1$
12. $\alpha^{12} \equiv \alpha^3$
13. $\alpha^{13} \equiv \alpha^2$
14. $\alpha^{14} \equiv \alpha$
15. $\alpha^{15} \equiv 1$

$\alpha^{15}=1$ 且沒有小於 15 的整數 k 使得 $\alpha^k=1$。
因為 α 是 $f_2(x)$ 的一個根，$\alpha^4=\alpha+1$
proof:

1. $\alpha^1 \equiv \alpha$

2. $\alpha^2 \equiv \alpha^2$
3. $\alpha^3 \equiv \alpha^3$
4. $\alpha^4 \equiv \alpha + 1$
5. $\alpha^5 \equiv \alpha^2 + \alpha$
6. $\alpha^6 \equiv \alpha^3 + \alpha^2$
7. $\alpha^7 \equiv \alpha^3 + \alpha + 1$
8. $\alpha^8 \equiv \alpha^3 + \alpha^2 + \alpha$
9. $\alpha^9 \equiv \alpha^3 + \alpha^2 + \alpha + 1$
10. $\alpha^{10} \equiv \alpha^3 + \alpha^2 + 1$
11. $\alpha^{11} \equiv \alpha^3 + \alpha$
12. $\alpha^{12} \equiv \alpha^3 + 1$
13. $\alpha^{13} \equiv \alpha^3$
14. $\alpha^{14} \equiv \alpha^2$
15. $\alpha^{15} \equiv 1$

Real-world Application

應用：LFSR（線性反饋移位暫存器）在流密碼中的應用
背景：
LFSR（線性反饋移位暫存器）是一種常見的流密碼生成器。它基於一個多項式來決定序列的生成方式。如果這個多項式是 本原多項式，則它能夠產生 最大長度序列（Maximum-Length Sequence, M-Sequence），即擁有 $2^n - 1$ 的週期長度，其中 n 是 LFSR 的位元數。

為何本原性至關重要：

1. 最大長度序列：

- 若 LFSR 的多項式是本原的，那麼它能夠產生 $2^n - 1$ 的序列長度，而不會出現較短的重複週期。
- 這確保了流密碼在一個極長的週期內保持高隨機性。

2. 均勻性與隨機性：

- 本原多項式生成的序列在 0 和 1 之間的分佈非常均勻，具有良好的統計性質。
- 這種隨機性對於密碼學安全至關重要。

3. 抗攻擊性：

- 若 LFSR 使用的多項式不是本原的，那麼產生的序列將具有明顯的重複性，攻擊者可以更容易地分析和破解這種序列。
- 本原多項式保證了最長周期，攻擊者無法輕易預測或破解序列。

實際例子：

在 A5/1 流密碼（早期 GSM 手機通信加密）中，LFSR 就是基於本原多項式來構造的。這些 LFSR 的本原性保證了產生的流密碼難以被預測和破解。