Irving Reyes Bravo

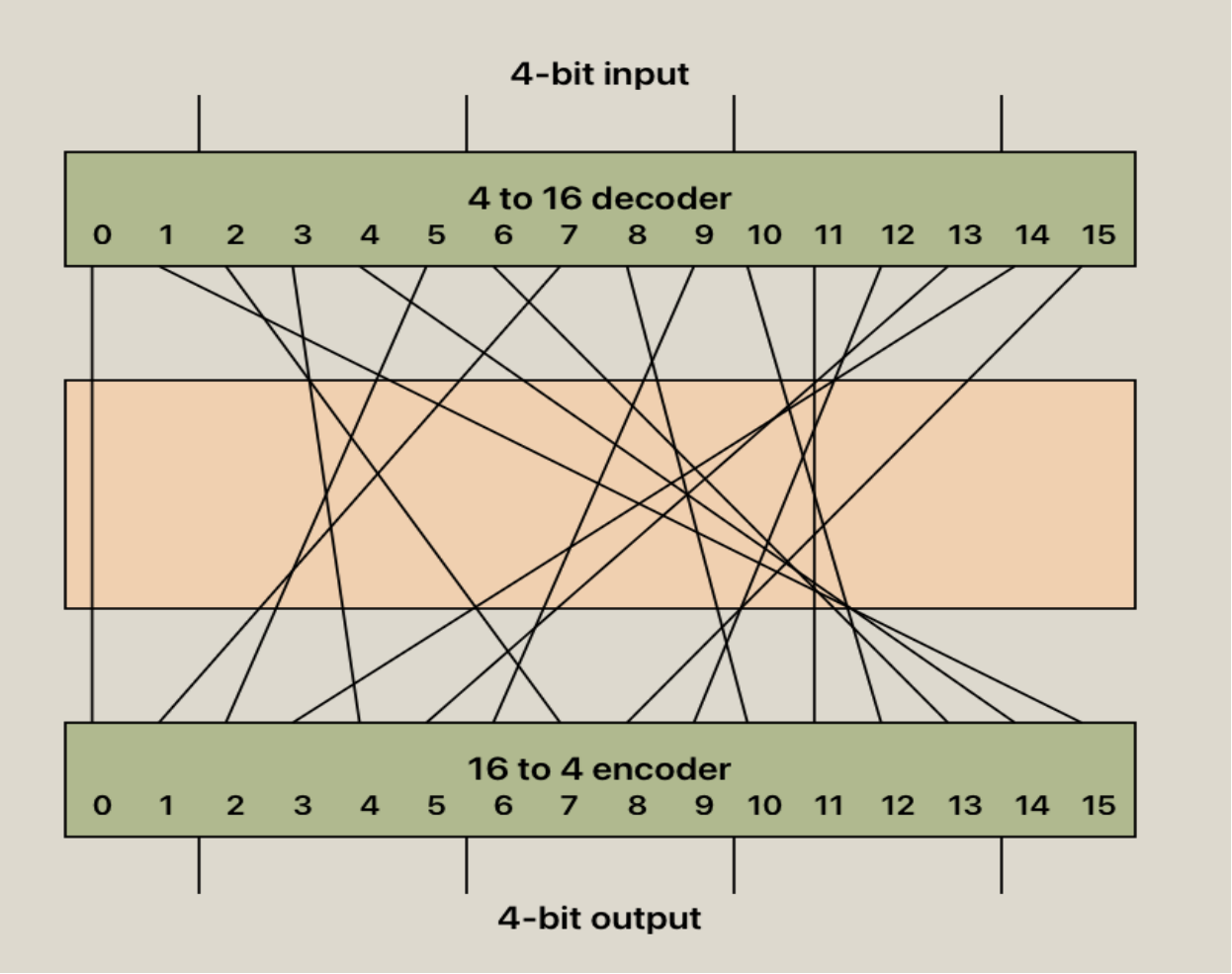
03/17/2025

CS 4920 Spring 2025

HW 3

Problem 1.

a. Consider the substitution defined by the third row (0yyyy1) of S-box S_1 in Slide 34 in the Ch.4 slide deck for Block Ciphers and DES slide deck. Draw and show a block diagram similar to Figure 4.2 in the textbook that corresponds to this substitution.



[made using *Canva*]

b. Compute the bits number 1, 16, 33, and 48 (i.e., the 1st bit, the 16th bit, the 33rd bit, and the 48th bit) at the output of the first round of the DES decryption. Assume that the ciphertext block is composed of all bit ones and the external key (the 64-bit key before the round subkey generations) is composed of all ones.
GIVEN:
Ciphertext is all ones (0xFFFFFFFFFFFFFFFF)
External Key is all ones (0xFFFFFFFFFFFFFFFF)
PROOF:
For the DES function (f):
    1. For a 64-bit all-ones key, I have a 56-bit effect all-one key (parity bits are dropped).
    2. K16 is generated (after key scheduling).
    3. The ciphertext goes through the initial permutations (IP).
    4. After IP, the block is split into left (L0) and right (R0) halves.
    5. In the first round of decryption: L1 = R0 and R1 = L0 $\oplus$ f(R0, K16):
        a. The 32-bit R0 is expanded to 48 bits.
        b. This is XORed with the 48-bit K16.
        c. The result is divided into eight 6-bit blocks.
        d. S-box S1 handles the first 6 bits:
        Based on the S-box S1:
            - **Bit 1 would be 0** (derived from S-box output)
            - **Bit 16 would be 1** (preserved from R0)
            - **Bit 33 would be 1** (preserved from L0)
            - **Bit 48 would be 0** (derived from S-box output)

Problem 2.
This problem provides a numerical example of encryption using a one-round version of DES. We start with the same pattern for key *K* and the plaintext.

GIVEN:

Hexadecimal notation: 8 9 A B C D E F 0 1 2 3 4 5 6 7

Binary notation: 1000 1001 1010 1011 1100 1101 1110 1111

0000 0001 0010 0011 0100 0101 0110 0111

I divide the plaintext into L0 and R0:

- L0 = 1000 1001 1010 1011 1100 1101 1110 1111
- R0 = 0000 0001 0010 0011 0100 0101 0110 0111

PROOF:

a. Derive K_1, the first-round subkey.
   Using the DES key scheduling algorithm (PC-1, left shifts, PC-2), I get:
   K1: 000001000100001101110110100100111001010110001101
b. Expand R_0 to get E[R_0], where E[·] is the expansion function.
   Using the expansion function:
   E[R0]: 100000000010100100000110101000001010101100001110
c. Calculate $A = \text{E}[R_0] \oplus K_1$.
   XOR operation:
   A: 100001000110101001110000001100110011111010000011
d. Apply S-box substitutions. Concatenate the results of (e) to get a 32-bit result, B.
   Splitting A into 8 6-bit segments and applying the corresponding S-box transformations:
   B (S-Box Output, 32-bit): 11111110011011111011111001011111
e. Apply the permutation to get P(B).
   P(B): 00101101010011010110100010110001
f. Calculate $\text{R}_1 = \text{P}(B) \oplus L_0$.
   XOR operation:
   R1: 10100100111001101010010101011110
g. Compute ciphertext:
   Ciphertext after one round is (R0, R1):
   Ciphertext:
   00000001001000110100010101100111 10100100111001101010010101011110
   Hexadecimal Ciphertext: 01234567 A49CA55E

Problem 3.

a. Does the set of residue classes (mod 4) form a group:
   i) with respect to modular addition?
   GIVEN: Set of residue classes mod 4 is Z4 = {0, 1, 2, 3}
   PROOF: To check if Z_4 forms a group under mod addition, I verify the group axioms:
   1. Closure: For any $a, b \in Z_4$, $a + b$ mod 4 is always in Z_4.
   2. Associativity: $(a + b) + c \equiv a + (b + c)$ mod 4, for all a, b, c.
   3. Identity: The identity element is 0, since $a + 0 \equiv a$ mod 4.
   4. Inverses:
      a. $0^{-1} = 0$,
      b. $1^{-1} = 3$ since $1 + 3 \equiv 0$ mod 4,
      c. $2^{-1} = 2$ since $2 + 2 \equiv 0$ mod 4,
      d. $3^{-1} = 1$.

   So, (Z_4, +) forms a group.

ii) with respect to modular multiplication?

GIVEN: The nonzero elements in Z4 are {1, 2, 3}

PROOF: To check if Z_4 forms a group under mod multiplication, I verify the group axioms:

1. Closure: For any $a, b \in Z_4$, $(a * b)$ mod 4 should be in {1, 2, 3}. However,
   2(2) = 4 ≡ 0 mod 4,
   which is not in the set.
2. Identity: The multiplicative identity is 1 since a * 1 = a mod 4.
3. Inverses:
   a. $1^{-1} = 1$,
   b. 2 has no inverse since 2x ≡ 1 mod 4 has no solution,
   c. $3^{-1} = 3$ since 3 * 3 = 9 ≡ 1 mod 4.

Since 2 lacks an inverse, Z_4^x is not a group under multiplication.

b. Does the set of residue classes (mod 4) form a field? Explain why.

Z_4 is a field only if:

- Z_4 is a commutative ring:
  - Since addition and multiplication are associative and commutative.
- Z_4 has a multiplicative identity of 1.
- Not every nonzero element has an inverse (i.e. 2).

Thus, Z_4 is not a field.

c. Develop a set of tables similar to Table 5.1 in the textbook for GF(5).

GF(5) is a finite field that consists of {0, 1, 2, 3, 4} with arithmetic module 5.

Addition Table (mod 5):

+	**0**	**1**	**2**	**3**	**4**
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

Multiplication Table (mod 5):

*	**0**	**1**	**2**	**3**	**4**
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Since every nonzero element has an inverse, GF(5) is a field.

d. Determine if the following are reducible over GF(2), i.e., coefficients mod 2.
A polynomial is irreducible if it cannot be factored into lower-degree polynomials in GF(2).

i) x^3 - 1

Factoring in GF(2): (x3 – 1)

= (x – 1)(x2 + x + 1)

Since (x2 + x + 1) has no roots in GF(2) , it is irreducible.

ii) x^3 - x^2 -1

Factoring in GF(2): (x3 – x2-1)

= (x2 + 1) + x2 + x

= 2x2 + x + 1

(x + 1)

Since no degree-1 factor divides it, and it doesn't factor, so it's irreducible.

iii) x^4 + 1

Factoring in GF(2): (x4 + 1)

= (x2 + 1)(x2 + 1) mod 2

Since (x2 + 1) is reducible over GF(2) – factors into (x+1)2, it is reducible.

Problem 4.

a. Develop a set of tables similar to Table 5.2 for GF(2^2) with $m(x) = x^2 + 1$.
Addition:

	+	**00**	**01**	**10**	**11**
00	0	0	1	x	(x + 1)
01	1	1	0	(x + 1)	x
10	x	x	(x + 1)	0	1
11	(x + 1)	(x + 1)	x	1	0

Multiplication:

	*	**00**	**01**	**10**	**11**
00	0	0	0	0	0
01	1	0	1	x	(x + 1)
10	x	0	x	1	(x + 1)
11	(x + 1)	0	(x + 1)	(x + 1)	1

Inverses:

w	**-w**	**w^{-1}**
0	0	-
1	1	1
x	x	x
(x + 1)	(x + 1)	(x + 1)

b. Develop a table similar to Table 5.5 in the textbook for GF(2^4) with $m(x) = x^4 + x + 1$.

Power Representation	**Polynomial Representation**	**Binary Representation**	**Decimal (Hex) Representation**
0	0	0000	0
g^0	1	0001	1
g^1	g^1	0010	2
g^2	g^2	0100	4
g^3	g^3	1000	8
g^4	g + 1	0011	3
g^5	g^2 + g	0110	6
g^6	$g^3 + g^2$	1100	c
g^7	g^3 + g + 1	1011	b
g^8	g^2 + 1	0101	5
g^9	g^3 + g	1010	a
g^{10}	g^2 + g + 1	0111	7
g^{11}	$g^3 + g^2$ + g	1110	e
g^{12}	$g^3 + g^2$ + g + 1	1111	f
g^{13}	$g^3 + g^2$ + 1	1101	d
g^{14}	g^3 + 1	1001	9

Problem 5. Finite-Field Calculator Programming

Implement a simple four-function calculator in GF(2^8) using the irreducible polynomial used in AES, i.e., $m(x)=x^8+x^4+x^3+x+1$. The four functions are addition, subtraction, multiplication, and division (where division is multiplication by the multiplicative inverse).

Here is the approach I used to implement a calculator for $GF(2^8)$ using the AES irreducible polynomial, which I represent as "0x11b" in Hexadecimal:

- **Representation:**
  Each element in $GF(2^8)$ is represented as an integer between 0 and 255, corresponding to the binary coefficients of a polynomial of degree 7 at most.
- **Operations:**
  - Addition/Subtraction:
    In $GF(2^n)$, addition and subtraction are the same operation – the bitwise XOR (^).
  - Multiplication:
    Implementation is followed by modular reduction via the AES polynomial.
  - Division:
    Implemented as multiplication by the multiplicative inverse.
  - Inverse:
    Uses the Extended Euclidean Algorithm to find multiplicative inverses.
- **Input/Output Handling:**
  My program first reads-in and converts binary strings to integers; then, after the calculations are performed, it converts the results back to their original form.

My program processes both the provided class-common inputs and my own calculations file. Below is a description of the main functions and variables within the code:

- **add_GF(a, b):**
  Performs addition using XOR.
- **subtract_GF(a, b):**
  Same as addition in this field.
- **multiply_GF(a, b):**
  Performs multiplication with reduction using the AES polynomial.
- **inverse_GF(a, b):**
  Computes multiplicative inverse using Extended Euclidean Algorithm.
- **divide_GF(a, b):**
  Performs division by multiplying with the inverse.
- **perform_calculation(line):**
  Processes a single calculation from the passed input file.
- **process_files(input, output):**
  Processes input file; writes calculation results to output file.
- **AES_ POLYNOMIAL:**
  Value "0x11b" representing $(x^8 + x^4 + x^3 + x + 1)$.

My program covers all four operations, with the test cases built to handle edge cases like division by zero (displays error messages). My program computes all operations on the fly *without* relying on precomputed tables.