МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КРЕМЕНЧУЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ МИХАЙЛА ОСТРОГРАДСЬКОГО
КАФЕДРА АВТОМАТИЗАЦІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ЗВІТ

ЗВІТ З ЛАБОРАТОРНИХ РОБІТ
З НАВЧАЛЬНОЇ ДИСЦИПЛІНИ
«ТЕХНОЛОГІЇ ЗАХИСТУ ІНФОРМАЦІЇ»

Виконав студент групи КН-23-1
Полинько Ігор Миколайович
Перевірив: ассистент кафедри АІС Андреєв П. І.

Кременчук 2025

ЛАБОРАТОРНА РОБОТА № 5

Тема: Шифрування алгоритмом DES

Мета: навчитися створювати програми шифрування (дешифрування) алгоритмом DES.

Порядок виконання роботи:

Реалізувати алгоритм DES (шифрування та дешифрування):

7–8 Робочий режим зворотний зв'язок за виходом OFB (Output Feed Back).

Варіант: 15

Скрипт програми:

```
from Crypto.Cipher import DES
import os

def bytes_to_bits(b: bytes) -> list:
    bits = []
    for byte in b:
        bits.extend([ (byte >> i) & 1 for i in range(7, -1, -1) ])
    return bits

def bits_to_bytes(bits: list) -> bytes:
    if len(bits) % 8 != 0:
        bits = bits + [0] * (8 - (len(bits) % 8))
    out = bytearray()
    for i in range(0, len(bits), 8):
        byte = 0
        for bit in bits[i:i+8]:
            byte = (byte << 1) | bit
        out.append(byte)
    return bytes(out)

def string_to_bits(s: str) -> list:
    return bytes_to_bits(s.encode('utf-8'))

def bits_to_string(bits: list) -> str:
    b = bits_to_bytes(bits)
    return b.decode('utf-8', errors='replace')

def xor_bits(a: list, b: list) -> list:
    n = min(len(a), len(b))
    return [ (a[i] ^ b[i]) for i in range(n) ]

def simple_des_block_encrypt(block_bits: list, key: bytes) -> list:
    block_bytes = bits_to_bytes(block_bits[:64])
    cipher = DES.new(key, DES.MODE_ECB)
    encrypted_bytes = cipher.encrypt(block_bytes)
    return bytes_to_bits(encrypted_bytes)

def des_ofb_encrypt(plaintext: str, key: bytes, iv: bytes) -> (bytes, bytes):
    pt_bits = string_to_bits(plaintext)
    B = 64
    prev = bytes_to_bits(iv)
    cipher_bits = []
```

```
    for i in range(0, len(pt_bits), B):
        block = pt_bits[i:i+B]
        keystream = simple_des_block_encrypt(prev, key)
        ks_slice = keystream[:len(block)]
        cipher_block = xor_bits(block, ks_slice)
        cipher_bits.extend(cipher_block)
        prev = keystream

    cipher_bytes = bits_to_bytes(cipher_bits)
    return iv, cipher_bytes

def des_ofb_decrypt(cipher_bytes: bytes, key: bytes, iv: bytes) -> str:
    cipher_bits = bytes_to_bits(cipher_bytes)
    B = 64
    prev = bytes_to_bits(iv)
    plain_bits = []

    for i in range(0, len(cipher_bits), B):
        block = cipher_bits[i:i+B]
        keystream = simple_des_block_encrypt(prev, key)
        ks_slice = keystream[:len(block)]
        plain_block = xor_bits(block, ks_slice)
        plain_bits.extend(plain_block)
        prev = keystream

    return bits_to_string(plain_bits)

if __name__ == "__main__":
    key = b'12345678'
    iv = os.urandom(8)

    plaintext = "Це тестовий текст для шифрування DES OFB"

    print("Вихідний текст:", plaintext)
    iv_out, ciphertext = des_ofb_encrypt(plaintext, key, iv)
    print("IV (hex):", iv_out.hex())
    print("Шифротекст (hex):", ciphertext.hex())

    recovered = des_ofb_decrypt(ciphertext, key, iv_out)
    print("Розшифрований текст:", recovered)
```

Результат:

```
Вихідний текст: Це тестовий текст для шифрування DES OFB
IV (hex): 3fb92780128c3d72
Шифротекст (hex): 0d2227a8637163fd379b9c0ceb0556ab4940545553 6fa410de5a3fe7eb2bc2a17a05adc256e87453b47207dcc57d9cbbcf98176adbcb1bb05da01b40cd6cdcd3c66bf2c6
Розшифрований текст: Це тестовий текст для шифрування DES OFB
```

Рисунок 5.1 – Результат роботи програми

Висновок: на цій лабораторній роботі ми навчилися створювати програми шифрування (дешифрування) алгоритмом DES.

Контрольні питання:

1. Структура алгоритму DES

Алгоритм DES є блочним симетричним шифром.

– Довжина блока: 64 біти, ключа — 56 біт.

– Процес складається з:

1) Початкової перестановки (IP);

2) 16 раундів за схемою Фейстеля:

– поділ блока на півблоки L і R;

– у кожному раунді:

L_i = R_{i-1},

R_i = L_{i-1} XOR F(R_{i-1}, K_i);

3) Кінцевої перестановки (IP^{-1}).

Розшифрування виконується аналогічно, але підключі використовуються у зворотному порядку.

2. Функція шифрування F

Функція F(R, K) виконує:

1) Розширення (E): 32 біти → 48 біт;

2) XOR з підключем K;

3) S-перетворення: 8 S-блоків по 6→4 біти;

4) P-перестановка: перестановка 32 біт за таблицею.

Результат — 32 біти.

3. Алгоритм обчислення підключів

1) Ключ 64 біти → PC-1 → 56 біт (видалення бітів парності).

2) Поділ на C і D (по 28 біт).

3) Для кожного з 16 раундів:

– циклічний зсув вліво на 1 або 2 біти;

– PC-2 – формування підключу K_i (48 біт).

У розшифруванні ключі застосовуються у зворотному порядку.

4. Зв’язок IP і IP^{-1}

Матриця IP^{-1} є оберненою до IP.

Застосування IP, а потім IP^{-1} повертає початковий порядок бітів: $IP^{-1}(IP(\text{дані})) = \text{дані}$.

5. Режим ECB (Electronic Code Book)

Кожен блок шифрується незалежно одним ключем:

C_i = E_K(P_i)

Недолік — однакові блоки породжують однакові шифроблоки.

6. Режим CBC (Cipher Block Chaining)

Кожен блок перед шифруванням XOR’иться з попереднім шифроблоком:

C_i = E_K(P_i XOR C_{i-1}), C_0 = IV.

Підвищує стійкість за рахунок залежності від попередніх блоків.

7. Режим CFB (Cipher Feedback)

Шифрується вхідний регістр (IV), результат XOR’иться з відкритим текстом: C_i = P_i XOR E_K(ShiftReg)

Регістр оновлюється частиною шифротексту.

Підходить для потокових даних.

8. Режим OFB (Output Feedback)

Вихід шифру подається назад на вхід: O_i = E_K(O_{i-1}), C_i = P_i XOR O_i.

Помилки передачі не накопичуються, бо зворотного зв’язку по шифротексту немає.