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 № 8

Тема: Алгоритм шифрування даних IDEA

Мета: вивчити алгоритм шифрування IDEA.

Порядок виконання роботи:

1. Реалізувати програмним способом 1–4 операції циклу алгоритму IDEA у режимі шифрування. Оформити програму у вигляді процедур і функцій.

2. Реалізувати програмним способом 5–9 операцій циклу алгоритму IDEA у режимі шифрування. Оформити програму у вигляді процедур і функцій.

3. Реалізувати програмним способом 10–14 операцій циклу алгоритму IDEA у режимі шифрування. Оформити програму у вигляді процедур і функцій.

4. Реалізувати програмним способом заключне перетворення виходу алгоритму IDEA у режимі шифрування. Оформити програму у вигляді процедур і функцій.

5. Використовуючи програми завдань 1–4, змоделювати програмним способом алгоритм IDEA у режимі шифрування.

Варіант: 15

Скрипт програми:

```
        import struct

def mul(a, b):
    return (a * b) % 0x10001 if a != 0 else (b % 0x10001)

def add(a, b):
    return (a + b) % 0x10000

def step1_to_4(X1, X2, X3, X4, K1, K2, K3, K4):
    Y1 = mul(X1, K1)
    Y2 = add(X2, K2)
    Y3 = add(X3, K3)
    Y4 = mul(X4, K4)
    return Y1, Y2, Y3, Y4

def step5_to_9(Y1, Y2, Y3, Y4, K5, K6):
    Z1 = Y1 ^ Y3
    Z2 = Y2 ^ Y4
    Z3 = mul(Z1, K5)
    Z4 = add(Z2, Z3)
    Z5 = mul(Z4, K6)
    Z6 = add(Z3, Z5)
```

```
    return Z3, Z4, Z5, Z6

def step10_to_14(Y1, Y2, Y3, Y4, Z3, Z4, Z5, Z6):
    X1_new = Y1 ^ Z5
    X2_new = Y3 ^ Z5
    X3_new = Y2 ^ Z6
    X4_new = Y4 ^ Z6
    return X1_new, X2_new, X3_new, X4_new

def final_transformation(X1, X2, X3, X4, K1, K2, K3, K4):
    Y1 = mul(X1, K1)
    Y2 = add(X2, K2)
    Y3 = add(X3, K3)
    Y4 = mul(X4, K4)
    return Y1, Y2, Y3, Y4

# --- IDEA шифрування одного блока ---
def IDEA_encrypt(block, round_keys):
    X1, X2, X3, X4 = block
    for i in range(8):
        K1, K2, K3, K4, K5, K6 = round_keys[i*6:(i+1)*6]
        Y1, Y2, Y3, Y4 = step1_to_4(X1, X2, X3, X4, K1, K2, K3, K4)
        Z3, Z4, Z5, Z6 = step5_to_9(Y1, Y2, Y3, Y4, K5, K6)
        X1, X2, X3, X4 = step10_to_14(Y1, Y2, Y3, Y4, Z3, Z4, Z5, Z6)
        if i != 7:
            X2, X3 = X3, X2
    Kf = round_keys[48:52]
    return final_transformation(X1, X2, X3, X4, *Kf)

# --- Генерація простого набору підключів (для тесту, без циклічного зсуву) ---
def generate_round_keys(key128):
    #  Приймаємо 128-бітний ключ як список 8 чисел по 16 біт
    round_keys = []
    for i in range(8):  # 8 раундів
        # беремо по 6 підключів на раунд, просто повторюючи ключові числа
        round_keys.extend([key128[(i+j)%8] for j in range(6)])
    # 4 ключі для фінального перетворення
    round_keys.extend(key128[:4])
    return round_keys

# --- Робота з файлами ---
def read_message_file(filename):
    with open(filename, "r") as f:
        text = f.read().strip()
    # перетворюємо символи на числа (0-9)
    return [int(c) for c in text]

def write_block_file(filename, blocks):
    with open(filename, "w") as f:
        for block in blocks:
            f.write(" ".join(str(x) for x in block) + "\n")

def main():
    message = read_message_file("message.txt")  # повідомлення у цифрах 0-9
    key128 = [0x1234,0x2345,0x3456,0x4567,0x5678,0x6789,0x7890,0x8901]  # приклад
ключа
    round_keys = generate_round_keys(key128)

    # розбиваємо повідомлення на блоки по 4 числа
    blocks = []
    for i in range(0, len(message), 4):
        block = message[i:i+4]
        while len(block) < 4:
            block.append(0)  # доповнення нулями
        blocks.append(tuple(block))
```

```
    # шифрування
    encrypted_blocks = [IDEA_encrypt(b, round_keys) for b in blocks]

    write_block_file("cipher.txt", encrypted_blocks)
    print("Шифрування завершено, результат у cipher.txt")

if __name__ == "__main__":
    main()
```

Результат програми:

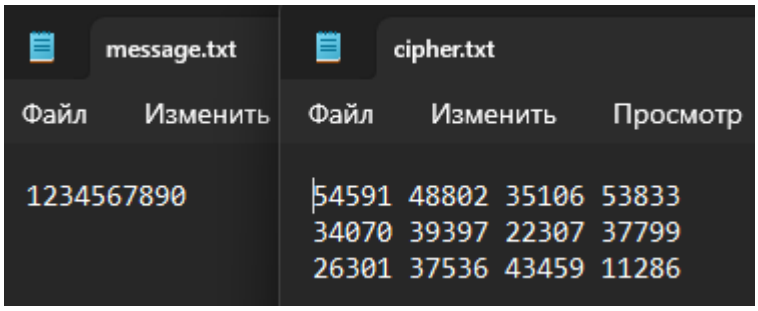

Рисунок 8.1 – Результат роботи програми

Висновок: на цій лабораторній роботі ми алгоритм шифрування IDEA. Використовуючи різні операції циклів реалізували алгоритм IDEA у режимі шифрування.

Контрольні питання:

1. Призначення та опис алгоритму IDEA

IDEA (International Data Encryption Algorithm) – це блочний симетричний шифр. Він працює з блоками по 64 біти та використовує ключ довжиною 128 біт. Алгоритм складається з 8 основних раундів та фінального перетворення. Кожен раунд містить операції: мультиплікативне та адитивне перетворення, операцію XOR і перестановку субблоків. Мета IDEA – забезпечити високий рівень стійкості до криптоаналізу при відносній простоті реалізації.

2. Призначення та опис алгоритму ДСТУ 28147

ДСТУ 28147 – український державний стандарт симетричного шифрування, блочний алгоритм із блоками 64 біти та ключем 256 біт. Складається з 32 раундів, кожен раунд використовує S-блоки для нелінійного перетворення та операції XOR і

перестановки. Використовується для захисту інформації в державних і корпоративних системах.

3. Створення псевдовипадкових чисел

Псевдовипадкові числа – це послідовності чисел, які виглядають випадковими, але створюються детермінованим способом. Найпоширеніші методи:

– лінійний конгруентний генератор (LCG);

– використання блочних шифрів у режимі лічильника (CTR);

– комбіновані генератори, де кілька методів поєднуються для підвищення якості.

4. Режими виконання алгоритму Blowfish

Blowfish – блочний симетричний шифр з блоком 64 біти. Можливі режими роботи:

– ECB (Electronic Codebook) – простий режим, кожен блок шифрується окремо;

– CBC (Cipher Block Chaining) – шифрування блоку залежить від попереднього;

– CFB (Cipher Feedback) – блочний шифр працює як потоковий;

– OFB (Output Feedback) – шифр генерує потокову маску незалежно від відкритого тексту;

– CTR (Counter) – блочний шифр використовує лічильник, паралельне шифрування можливе.

5. Режими виконання алгоритму IDEA

IDEA – блочний шифр з блоками 64 біти. Підтримує ті самі режимі, що і Blowfish:

– ECB – базовий режим без зв’язку між блоками;

– CBC – попередній блок впливає на поточний;

– CFB – дозволяє використовувати блочний шифр як потоковий;

– OFB – генерує потокову маску для кожного блоку;

– CTR – шифрування блоку через лічильник, дозволяє паралельне шифрування.