Міністерство освіти і науки України

Національний технічний університет «ХПІ»

Навчально-науковий інститут комп’ютерних наук та інформаційних технологій

Кафедра комп’ютерної інженерії та програмування

ЗВІТ

з лабораторної роботи № 2

з дисципліни «Сучасні технології безпечного програмування»

«**СИМЕТРИЧНЕ ШИФРУВАННЯ. АЛГОРИТМ AES**»

Виконав:

студент гр. КН-Н922б

Кулик Д.І.

Перевірив:

Бульба С. С.

Харків – 2022

Мета роботи: Дослідити принципи роботи симетричного шифрування на прикладі алгоритму AES.

Індивідуальне завдання

Реалізувати алгоритм симетричного шифрування AES (будь-якої версії - 128 або 256).

Довести коректність роботи реалізованого алгоритму шляхом порівняння результатів з існуючими реалізаціями (напр. сайтом-утилітою https://cryptii.com).

Хід роботи

Розширений стандарт шифрування (AES) — це симетричний блоковий шифр, обраний урядом США для захисту секретної інформації. AES реалізовано в програмному та апаратному забезпеченні по всьому світу для шифрування конфіденційних даних. Це має важливе значення для комп'ютерної безпеки уряду, кібербезпеки та захисту електронних даних.

Алгоритм має чотири трансформації, кожна з яких своїм чином впливає на стан State і зрештою призводить до результату: SubBytes(), ShiftRows(), MixColumns() та AddRoundKey(). Загальну схему шифрування можна представити як:

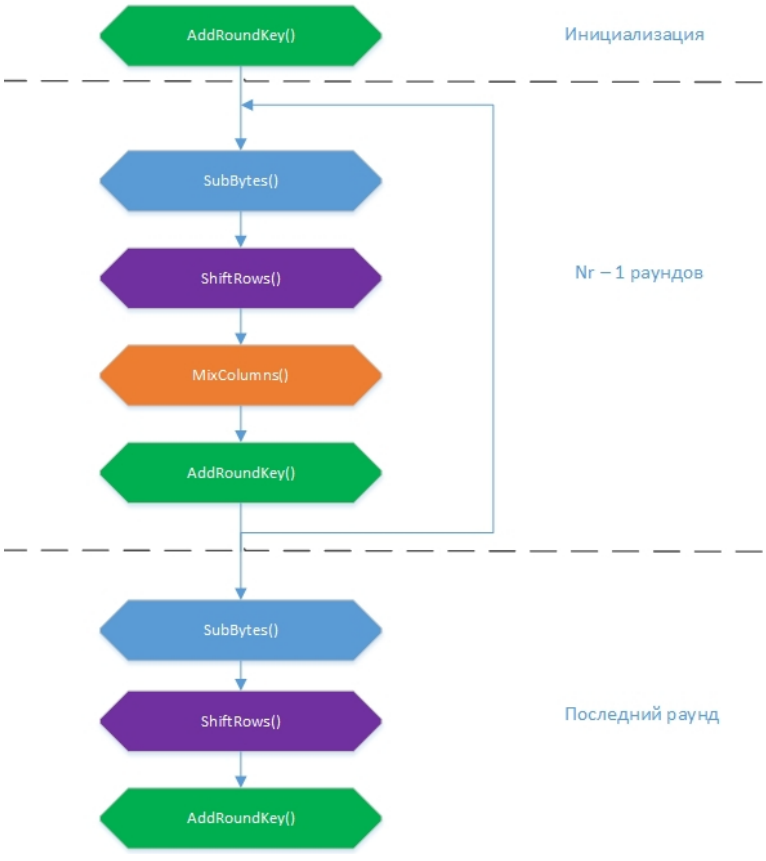

Рисунок 1 – Загальна схема шифрування

Алгоритм дешифрування простий: якщо з тим самим ключовим словом виконати послідовність трансформацій, інверсних трансформацій шифрування, то вийде вихідне повідомлення. Такими інверсними трансформаціями є InvSubBytes(), InvShiftRows(), InvMixColumns() та AddRoundKey(). Загальна схема алгоритму розшифрування:

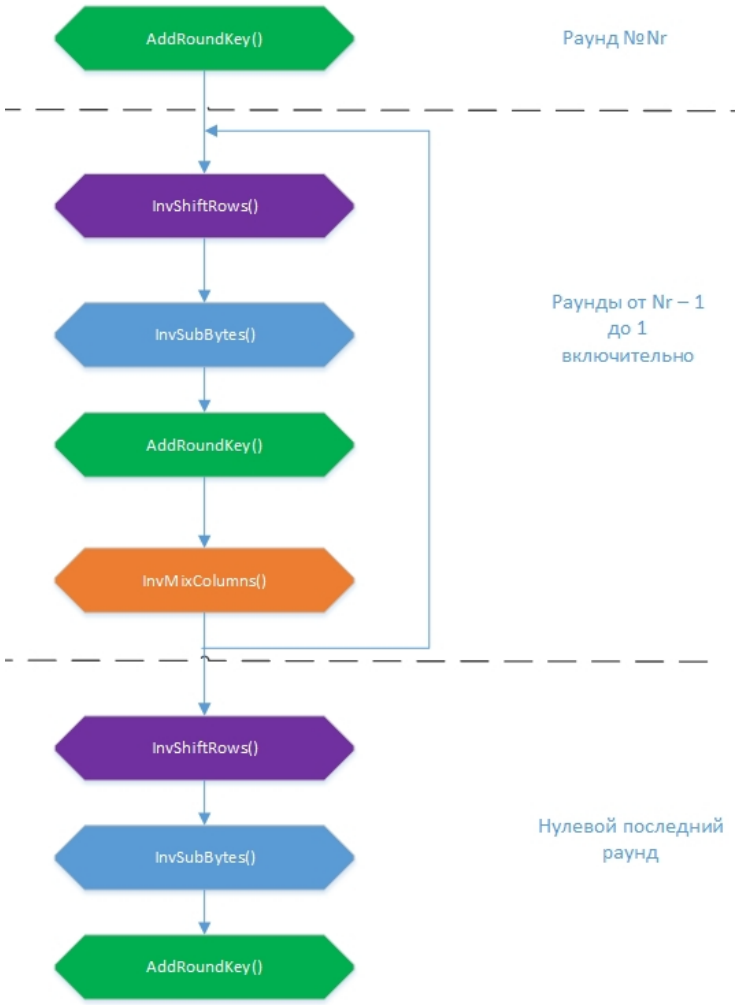

Рисунок 2 – Загальна схема дешифрування

Важливі фрагменти програми

```
def add_round_key(s, k):
    """Функція виконує побітовий XOR кожного елемента зі State з відповідним
    елементом з RoundKey, що будується для кожного раунду на основі секретного
    ключа функцією _expand_key()"""
    for i in range(4):
        for j in range(4):
            s[i][j] ^= k[i][j]
```

```
def sub_bytes(s):
    """Функція, що виконує заміну кожного байта з State матриці
    на відповідний йому із константної таблиці s_box"""
    for i in range(4):
        for j in range(4):
            s[i][j] = s_box[s[i][j]]
def shift_rows(s):
    """Функція, що виконує циклічний зсув вліво на 1 елемент для першого рядка,
    на 2 для другого і на 3 для третього. Нульовий рядок не зміщується."""
    s[0][1], s[1][1], s[2][1], s[3][1] = s[1][1], s[2][1], s[3][1], s[0][1]
    s[0][2], s[1][2], s[2][2], s[3][2] = s[2][2], s[3][2], s[0][2], s[1][2]
    s[0][3], s[1][3], s[2][3], s[3][3] = s[3][3], s[0][3], s[1][3], s[2][3]
def mix_columns(s):
    """Функція, що виконує дії функції mix_single_column() для кожної
    колонки в State"""
    for i in range(4):
        mix_single_column(s[i])
```

Рисунок 3 – Методи для шифрування

```
def inv_sub_bytes(s):
    """Функція працює так само, як і sub_bytes(), за винятком того,
    що заміни робляться з константної таблиці inv_s_box"""
    for i in range(4):
        for j in range(4):
            s[i][j] = inv_s_box[s[i][j]]

def inv_shift_rows(s):
    """Функція працює так само, як і shift_rows(), за винятком того,
    що циклічний зсув виконується вправо"""
    s[0][1], s[1][1], s[2][1], s[3][1] = s[3][1], s[0][1], s[1][1], s[2][1]
    s[0][2], s[1][2], s[2][2], s[3][2] = s[2][2], s[3][2], s[0][2], s[1][2]
    s[0][3], s[1][3], s[2][3], s[3][3] = s[1][3], s[2][3], s[3][3], s[0][3]

def inv_mix_columns(s):
    """Функція, що виконує ті ж самі операції, що й mix_columns(), але кожна колонка
    State перемножується з іншим многочленом {0b}x^3 + {0d}x^2 + {09}x + {0e}"""
    for i in range(4):
        u = xtime(xtime(s[i][0] ^ s[i][2]))
        v = xtime(xtime(s[i][1] ^ s[i][3]))
        s[i][0] ^= u
        s[i][1] ^= v
        s[i][2] ^= u
        s[i][3] ^= v

    mix_columns(s)
```

Рисунок 4 – Методи для дешифрування

```
def encrypt_block(self, plaintext):
    """..."""

    assert len(plaintext) == 16

    plain_state = bytes2matrix(plaintext)

    add_round_key(plain_state, self._key_matrices[0])

    for i in range(1, self.n_rounds):
        sub_bytes(plain_state)
        shift_rows(plain_state)
        mix_columns(plain_state)
        add_round_key(plain_state, self._key_matrices[i])

    sub_bytes(plain_state)
    shift_rows(plain_state)
    add_round_key(plain_state, self._key_matrices[-1])

    return matrix2bytes(plain_state)
```

Рисунок 5 – Реалізація шифрування

```
def encrypt_block(self, plaintext):
    """..."""

    assert len(plaintext) == 16

    plain_state = bytes2matrix(plaintext)

    add_round_key(plain_state, self._key_matrices[0])

    for i in range(1, self.n_rounds):
        sub_bytes(plain_state)
        shift_rows(plain_state)
        mix_columns(plain_state)
        add_round_key(plain_state, self._key_matrices[i])

    sub_bytes(plain_state)
    shift_rows(plain_state)
    add_round_key(plain_state, self._key_matrices[-1])

    return matrix2bytes(plain_state)
```

Рисунок 6 – Реалізація дешифрування

Результати роботи програми

```
C:\Users\Daniil\PycharmProjects\stbp\Scripts\python.exe C:/Users/Daniil/PycharmProjects/stbp/LABS/kulyk02/main.py
Заданий текст          :b'Daniil Kulyk'
Текст у байтах         :44616e69696c204b756c796b
Зашифрований текст     :b37913ec611d1d3acb6cf15d
Розшифрований текст    :b'Daniil Kulyk'
```

Рисунок 7 – Результат виконання програми

Рисунок 8 – Перевірка результату шифрування за допомогою сайту-утиліти https://cryptii.com

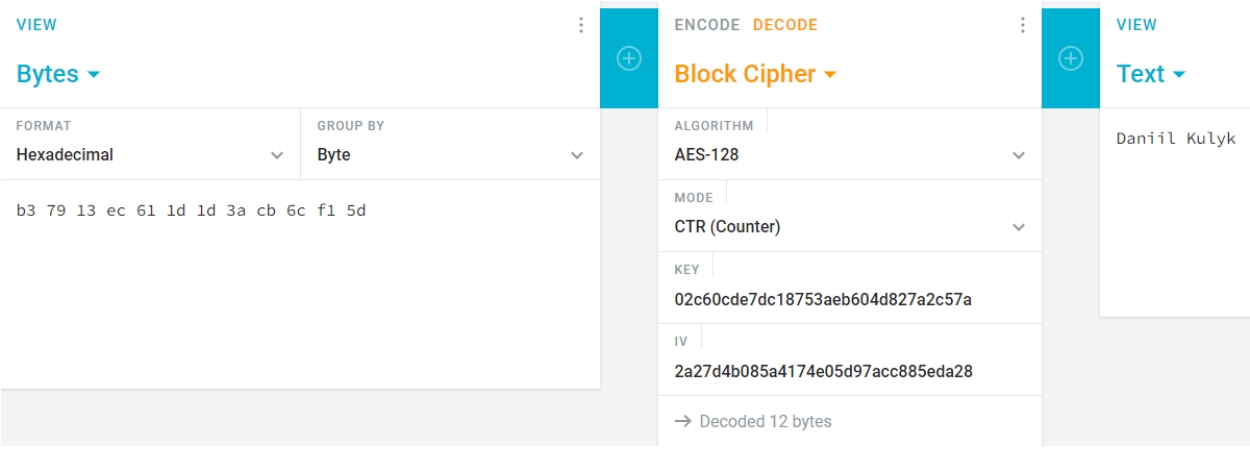

Рисунок 9 – Перевірка результату дешифрування за допомогою сайту-утиліти https://cryptii.com

При порівнянні можемо побачити що результат виконання реалізацій алгоритму однаковий.

Висновки: в результаті виконання лабораторної роботи було досліджено принципи роботи симетричного шифрування на прикладі алгоритму AES. В результаті порівняння власної реалізації алгоритму з вже реалізованими була виявлена ідентичність роботи, що доводить коректність першого.