MainPage/Info Security/Encript Lab 1.3

Университет ИТМО
Факультет ФПИ и КТ

Отчёт по лабораторной работе 1.3

«Поточное симметричное шифрование»

по дисциплину

«Информационная безопасность»

Студент: Чжоу Хунсян
Группа: P34131
Преподаватель:

Санкт-Петербург 2024

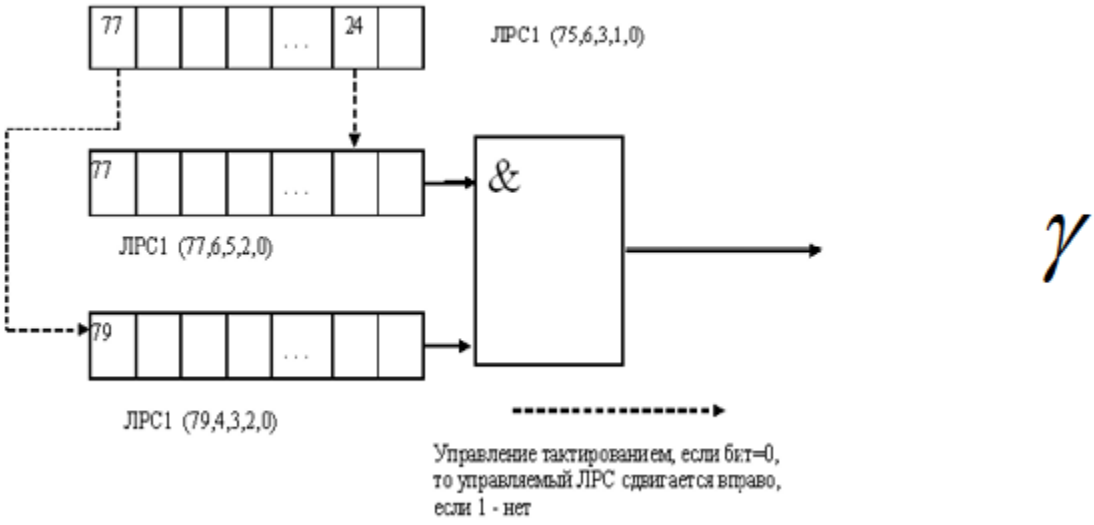
Цель работы

изучение структуры и основных принципов работы современных алгоритмов поточного симметричного шифрования, приобретение навыков программной реализации поточных симметричных шифров.

Варианты заданий

> Вариант: 23 % 10 = 3

Реализовать в программе поточное кодирование текста, вводимого с клавиатуры, с помощью заданной нелинейной схемы РС.



Исходный код

```
class Registers:
    # Initialize the shift register with a bit length, an initial value, and feedback posi
    def __init__(self, bit_length, initial_value, feedback_positions):
        self.bit_length = bit_length  # The number of bits in the register
        # Initialize the register, ensuring it is masked to the specified bit length
        self.register = initial_value & ((1 << bit_length) - 1)
        self.feedback_positions = feedback_positions  # Positions used for feedback in the

    # Perform a shift operation on the register and return the output bit
    def shift(self):
        output_bit = self.register & 1  # The least significant bit of the register is the
        feedback = 0  # Initialize the feedback value
        # XOR the bits at the specified feedback positions to compute the feedback
        for pos in self.feedback_positions:
            feedback ^= (self.register >> pos) & 1
        # Perform the shift operation: feedback bit is shifted into the most significant
        # and the remaining bits are shifted right by one
        self.register = (feedback << (self.bit_length - 1)) | (self.register >> 1) & ((1 <
        return output_bit  # Return the output bit

    # Return the least significant bit of the register (used for output generation)
    def get_output(self):
        return self.register & 1


# Initialize the shift registers with the given initial register value
def setRegisters(init_register):
    # Create three shift registers with different bit lengths and feedback positions
    lsr1 = Registers(78, init_register, [75, 6, 3, 1, 0])
    lsr2 = Registers(78, init_register, [77, 6, 5, 2, 0])
    lsr3 = Registers(80, init_register, [79, 4, 3, 2, 0])

    return lsr1, lsr2, lsr3  # Return the initialized shift registers


# Generate the output for encryption/decryption using the shift registers
def y_generation(lsr1, lsr2, lsr3):
    y = 0  # Initialize the output value
    for i in range(8):  # Iterate 8 times to generate a byte of output
```

```
        lsr1.shift()  # Shift the first shift register
        # Shift the second register if the 25th bit of the first register is 0
        if not (lsr1.register >> 24) & 1:
            lsr2.shift()
        # Shift the third register if the 78th bit of the first register is 0
        if not (lsr1.register >> 77) & 1:
            lsr3.shift()
        # Generate the output bit for this iteration by combining the outputs of the secon
        y |= lsr3.get_output() & lsr2.get_output() << i
    return y  # Return the generated output byte


# Encrypt the plaintext using the initial register value
def encrypt(plaintext, init_register):
    text_bytes = plaintext.encode("utf-8")  # Convert the plaintext to bytes
    return bytes_xor(text_bytes, init_register)  # Encrypt the bytes using XOR with the ge


# Decrypt the encrypted data by XORing it with the same sequence used during encryption
def decrypt(text_bytes, init_register):
    return bytes_xor(text_bytes, init_register).decode("utf-8")  # Decrypt and return as a


# Perform the XOR operation between the plaintext bytes and the pseudo-random sequence ge
def bytes_xor(text_bytes, init_register):
    # Initialize the shift registers with the given initial value
    lsr1, lsr2, lsr3 = setRegisters(init_register)
    result = bytearray()  # Initialize a bytearray to store the result
    for byte in text_bytes:
        y = y_generation(lsr1, lsr2, lsr3)  # Generate the pseudo-random output byte
        result.append(byte ^ y)  # XOR the current byte with the output and append to the
    return result  # Return the resulting encrypted/decrypted data


# Main function to run the encryption and decryption process
if __name__ == '__main__':
    # Initialize the starting value for the shift registers (in hexadecimal)
    init_register = 0xACACACACACACACACACAC
    print("Initial register: " + hex(init_register) + '\n')
```

```
# Read the plaintext from the input file
with open("input.txt", "r", encoding="utf-8") as f:
    text = f.read()
print("Plain Text:\n" + text + '\n')

# Encrypt the plaintext
encrypted = encrypt(text, init_register)
# Write the encrypted data to a file in binary format
with open("encrypted.txt", "wb") as f:
    f.write(encrypted)
print("Encrypted Text:\n" + encrypted.decode("utf-8") + '\n')

# Read the encrypted data back from the file
with open("encrypted.txt", "rb") as f:
    encrypted_data = f.read()

# Decrypt the encrypted data
decrypted = decrypt(encrypted_data, init_register)
# Write the decrypted data back to a file
with open("decrypted.txt", "w", encoding="utf-8") as f:
    f.write(decrypted)
print("Decrypted Text:\n" + decrypted + '\n')
```

Результаты работы программы

```
C:\Users\Tolia\AppData\Local\Programs\Python\Python310\python.exe C:\Users\Tolia\Documents
Initial register: 0xacacacacacacacacacac

Plain Text:
В рамках нашей работы мы определили следующую методологическую базу:
Сначала мы провели тщательный анализ существующих решений по дизайну интерфейса. Это включ
На основе результатов анализа мы разработали теоретическую модель. Эта модель включала кл
С помощью разработанной модели мы приступили к экспериментальному проектированию. Был созд
Заключительным этапом было тестирование созданного прототипа. Мы провели серию тестов с уч

Encrypted Text:
В Èанках нбJVи!cΨбпуъ мы oщpeVvoйки!рлvдующтЎ мVтЮдщлЮгицvÈлЃЎ абзЃ: CѾaIΨкб мъ щроθели ЂJ
ùа осноге!резуoьуψтоθ Ψнакиўб мы сазÈааотΨлй уеорvуσчдЁкую мпvеoЌ.!ѫЃΨ!нпдекь θклячакб!oOк
ЗΨккючйуеOьмъм ЙЂΨщом былп тдстировбние!cщўданнЮвщ псототσоа/ Мы псовели серйЎ тvсЂов с уч

Decrypted Text:
В рамках нашей работы мы определили следующую методологическую базу:
Сначала мы провели тщательный анализ существующих решений по дизайну интерфейса. Это включ
На основе результатов анализа мы разработали теоретическую модель. Эта модель включала кл
С помощью разработанной модели мы приступили к экспериментальному проектированию. Был созд
Заключительным этапом было тестирование созданного прототипа. Мы провели серию тестов с уч


Process finished with exit code 0
```

Вывод

В ходу лабораторной работы, я выучил несколько подходы для щифрования и дешифрования текст в файле.