МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИМЕНИ П. О. СУХОГО

Факультет автоматизированных и информационных систем

Кафедра «Информатика»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

по дисциплине **«Методы защиты информации»**

на тему: «**Стандарты симметричного шифрования DES и ГОСТ 28147-89**»

Выполнили: студент гр. ИП-41
Шевкунова В.А.
Коваленко А.И.
Принял: преподаватель
Бородин Н.Н.

Гомель 2023

Цель: получить практические навыки решения задач целочисленного программирования.

Задание I. Реализовать приложение для шифрования, позволяющее выполнять следующие действия: 1. Шифровать данные по заданному в варианте алгоритму: – шифруемый текст должен храниться в одном файле, а ключ шифрования – в другом; – зашифрованный текст должен сохраняться в файл; – в процессе шифрования предусмотреть возможность просмотра и изменения ключа, шифруемого и зашифрованного текстов в шестнадцатеричном и символьном виде; II. Реализовать приложение для дешифрования, позволяющее выполнять следующие действия: 2. Дешифровать данные по заданному в варианте алгоритму: – зашифрованный текст должен храниться в одном файле, ключ – в другом; – расшифрованный текст должен сохраняться в файл; – в процессе дешифрования предусмотреть возможность просмотра и изменения ключа, зашифрованного и расшифрованного текстов в шестнадцатеричном и символьном виде.

Вариант 3

№	Алгоритм шифрования	Режим работы
1	ГОСТ 28147-89	Режим простой замены
2	ГОСТ 28147-89	Гаммирование
3	ГОСТ 28147-89	Гаммирование с обратной связью
4	ГОСТ 28147-89	Выработка имитовставки

Листинг программы:

```
import binascii


def gost_encrypt(key, data):
    # Преобразование ключа в список байтов
    key_bytes = bytearray.fromhex(key)

    # Преобразование данных в список байтов
    data_bytes = bytearray(data.encode())

    # Гаммирование с обратной связью
    iv = bytearray(8)  # Инициализирующий вектор
    encrypted_data = bytearray()

    for i in range(len(data_bytes)):
        # Шифрование очередного байта данных
        encrypted_byte = data_bytes[i] ^ iv[i % 8] ^ key_bytes[i %
len(key_bytes)]
        encrypted_data.append(encrypted_byte)

        # Обновление инициализирующего вектора
        iv[i % 8] = encrypted_byte

    # Возвращение зашифрованных данных в виде списка байтов
    return encrypted_data


def gost_decrypt(key, data):
    # Преобразование ключа в список байтов
    key_bytes = bytearray.fromhex(key)
```

```
    # Гаммирование с обратной связью
    iv = bytearray(8)  # Инициализирующий вектор
    decrypted_data = bytearray()

    for i in range(len(data)):
        # Расшифровка очередного байта данных
        decrypted_byte = data[i] ^ iv[i % 8] ^ key_bytes[i % len(key_bytes)]
        decrypted_data.append(decrypted_byte)

        # Обновление инициализирующего вектора
        iv[i % 8] = data[i]

    # Возвращение расшифрованных данных в виде строки
    return decrypted_data.decode()

def save_bytes_to_file(filename, data):
    with open(filename, "wb") as f:
        f.write(data)


def save_string_to_file(filename, data):
    with open(filename, "w") as f:
        f.write(data)


def read_bytes_from_file(filename):
    with open(filename, "rb") as f:
        return f.read()


def read_string_from_file(filename):
    with open(filename, "r") as f:
        return f.read()


def print_hex_and_text(data, label):
    hex_data = binascii.hexlify(data).decode()
    text_data = data.decode()
    print(f"{label} (Hex): {hex_data}")
    print(f"{label} (Text): {text_data}")


def main():
    key_file = "key.txt"
    data_file = "data.txt"
    encrypted_file = "encrypted.txt"
    decrypted_file = "decrypted.txt"

    # Чтение ключа из файла
    key = read_string_from_file(key_file)

    # Чтение данных из файла
    data = read_string_from_file(data_file)

    # Вывод исходных данных
    print("Original Key and Data:")
    print_hex_and_text(key.encode(), "Key")
    print_hex_and_text(data.encode(), "Data")
    print()

    # Шифрование данных
    encrypted_data = gost_encrypt(key, data)
```

```
# Вывод зашифрованных данных
print("Encrypted Data:")
print_hex_and_text(encrypted_data, "Encrypted Data")
print()

# Сохранение зашифрованных данных в файл
save_bytes_to_file(encrypted_file, encrypted_data)

# Загрузка зашифрованных данных из файла
encrypted_data = read_bytes_from_file(encrypted_file)

# Дешифрование данных
decrypted_data = gost_decrypt(key, encrypted_data)

# Вывод расшифрованных данных
print("Decrypted Data:")
print_hex_and_text(decrypted_data.encode(), "Decrypted Data")
print()

# Сохранение расшифрованных данных в файл
save_string_to_file(decrypted_file, decrypted_data)

# Вывод успешного завершения операции
print("Encryption and decryption completed successfully.")

# Изменение ключа
new_key = input("Enter a new key: ")
key = new_key.strip()

# Изменение данных
new_data = input("Enter new data: ")
data = new_data.strip()

# Вывод обновленных данных
print("Updated Key and Data:")
print_hex_and_text(key.encode(), "Key")
print_hex_and_text(data.encode(), "Data")
print()

# Шифрование обновленных данных
encrypted_data = gost_encrypt(key, data)

# Вывод зашифрованных данных
print("Updated Encrypted Data:")
print_hex_and_text(encrypted_data, "Encrypted Data")
print()

# Сохранение обновленных зашифрованных данных в файл
save_bytes_to_file(encrypted_file, encrypted_data)

# Загрузка обновленных зашифрованных данных из файла
encrypted_data = read_bytes_from_file(encrypted_file)

# Дешифрование обновленных данных
decrypted_data = gost_decrypt(key, encrypted_data)

# Вывод расшифрованных обновленных данных
print("Updated Decrypted Data:")
print_hex_and_text(decrypted_data.encode(), "Decrypted Data")
print()

# Сохранение расшифрованных обновленных данных в файл
save_string_to_file(decrypted_file, decrypted_data)
```

```
    # Вывод успешного завершения операции
    print("Encryption and decryption of updated data completed success-
fully.")


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

Результат работы.

```
Original Key and Data:
Key (Hex): 353436353733373436623635373 9
Key (Text): 546573746b6579
Data (Hex): 48656c6c6f2c20776f726c6421205468697320697320736f6d6520646174612074 6f20656e63727970742e
Data (Text): Hello, world! This is some data to encrypt.

Encrypted Data:
Encrypted Data (Hex): 1c001f1804495923160107174010592e0c064c1b4a644f32150809067f755d660a02503774655b741f0f2a
Encrypted Data (Text): FS NUL US CAN89IY#SYN5@DLEY.FF ACKLESCJdO2 ACK DEL U]f
STXP7te[t=  =*

Decrypted Data:
Decrypted Data (Hex): 48656c6c6f2c20776f726c6421205468697320697320736f6d6520646174612074 6f20656e63727970742e
Decrypted Data (Text): Hello, world! This is some data to encrypt.

Encryption and decryption completed successfully.
Enter a new key: 546573746b6523
Enter new data: Anastasia Kovalenko
Updated Key and Data:
Key (Hex): 35343635373337343662363533233
Key (Text): 546573746b6523
Data (Hex): 416e617374617361204b6f76616c656e6b6f
Data (Text): Anastasia Kovalenko

Updated Encrypted Data:
Encrypted Data (Hex): 150b12071f04503d11582d030c46683d0c4729
Encrypted Data (Text): NAK VT DC2BEL US EOTP=DC1X-ETX FFFh=FFG)

Updated Decrypted Data:
Decrypted Data (Hex): 416e617374617361204b6f76616c656e6b6f
Decrypted Data (Text): Anastasia Kovalenko

Encryption and decryption of updated data completed successfully.

Process finished with exit code 0
```

Рисунок 1 – Результат выполнения программы

Вывод: при выполнении работы были получены практические навыки решения задач целочисленного программирования.