МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных систем и технологий

**Отчет**

**по лабораторной работе № 3**

**«Алгоритм шифрования данных DES»**

по дисциплине: «Основы информационной безопасности»

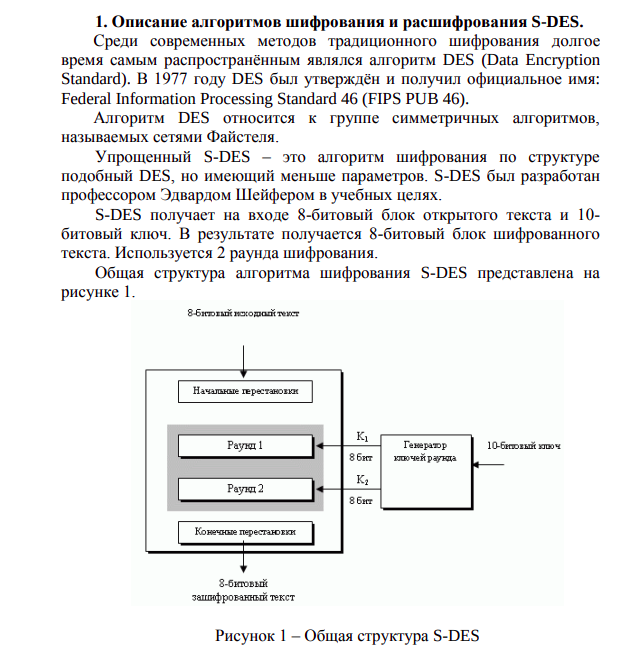
|  |  |
| --- | --- |
| Выполнили**:** | ст.гр. 10701322 |
|  | Махонина Ю.В.  Свиржевская В.В. |
|  |  |
|  |  |
| Принял: | Давыденко Н.В. |

Минск 2024

**Цель работы:**

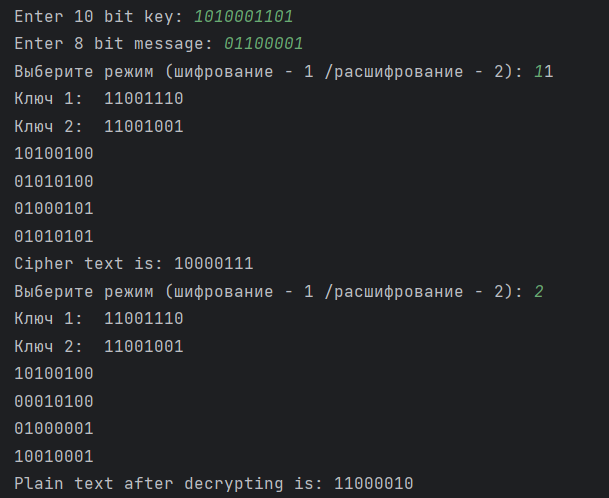
Познакомиться с основами симметричного шифрования. Изучить алгоритм шифрования DES на примере упрощенной версии S-DES.

**Описание алгоритма шифрования и расшифрования:**



**Задания на лабораторную работу**

1. Изучить основы симметричного шифрования 2. Изучить алгоритмы шифрования DES и S-DES. 3. Написать программы шифрования и расшифрования одного символа с использованием алгоритма S-DES. При шифровании/расшифровании ключ и символ вводить с клавиатуры в двоичном или десятичном виде (как значение ASCII-кода символа). Промежуточные значения выводить на экран в двоичном виде. Результат шифрования/расшифрования выводить на экран в двоичном или десятичном виде.



Результат выполнения задания

**Контрольные вопросы**

**Несингулярные преобразования** — это преобразования, которые имеют обратное. Пример несингулярного преобразования: поворот матрицы. Пример сингулярного преобразования: проекция на одну из осей.

**Структура сети Файстеля** включает в себя несколько раундов, каждый из которых состоит из следующих шагов:

* 1. Разделение блока данных на две части.
  2. Применение функции шифрования к одной из частей и сложение результата с другой частью.
  3. Перестановка частей местами.

**Классическая сеть Файстеля** — это сеть, в которой каждый раунд использует одну и ту же функцию шифрования. **Гомогенная сеть Файстеля** — это сеть, в которой все раунды идентичны по структуре и используют одинаковые функции.

**Расшифрование в сетях Файстеля** выполняется аналогично шифрованию, но с обратным порядком применения ключей.

**Криптоаналитическая стойкость шифра Файстеля** зависит от:

* 1. Количества раундов.
  2. Сложности и криптостойкости используемой функции шифрования.
  3. Длины ключа.

**Алгоритм DES** (Data Encryption Standard) является сетью Файстеля, так как он использует структуру сети Файстеля для шифрования данных. DES включает 16 раундов, каждый из которых использует уникальный подключ.

**Сравнение алгоритмов DES и S-DES**:

* 1. Длина ключа: DES — 56 бит, S-DES — 10 бит.
  2. Длина блока шифрования: DES — 64 бита, S-DES — 8 бит.
  3. Количество раундов: DES — 16, S-DES — 8.
  4. Количество подключей: DES — 16, S-DES — 8.
  5. Размер и количество S-блоков: DES — 8 S-блоков по 4 бита, S-DES — 2 S-блока по 4 бита.

**Криптостойкость алгоритма DES**: DES считается устаревшим и недостаточно стойким для современных требований безопасности из-за короткой длины ключа. На практике его заменили более стойкие алгоритмы, такие как AES.

**Типы операций в современных блочных алгоритмах симметричного шифрования**:

* 1. Перестановки.
  2. Подстановки.
  3. Линейные преобразования.
  4. Нелинейные преобразования.

**Листинг кода:**

**def apply\_table(inp, table):**

**res = ""**

**for i in table:**

**res += inp[i - 1]**

**return res**

**def left\_shift(data):**

**return data[1:] + data[0]**

**def xor(a, b):**

**res = ""**

**for i in range(len(a)):**

**if a[i] == b[i]:**

**res += "0"**

**else:**

**res += "1"**

**return res**

**def apply\_sbox(s, data):**

**row = int("0b" + data[0] + data[-1], 2)**

**col = int("0b" + data[1:3], 2)**

**return bin(s[row][col])[2:]**

**def function(expansion, s0, s1, key, message):**

**left = message[:4]**

**right = message[4:]**

**temp = apply\_table(right, expansion)**

**temp = xor(temp, key)**

**left\_bin\_str = apply\_sbox(s0, temp[:4])**

**right\_bin\_str = apply\_sbox(s1, temp[4:])**

**left\_bin\_str = "0" \* (2 - len(left\_bin\_str)) + left\_bin\_str**

**right\_bin\_str = "0" \* (2 - len(right\_bin\_str)) + right\_bin\_str**

**temp = apply\_table(left\_bin\_str + right\_bin\_str, p4\_table)**

**temp = xor(left, temp)**

**return temp + right**

**if \_\_name\_\_ == "\_\_main\_\_":**

**key = input("Enter 10 bit key: ")**

**message = input("Enter 8 bit message: ")**

**p8\_table = [6, 3, 7, 4, 8, 5, 10, 9]**

**p10\_table = [3, 5, 2, 7, 4, 10, 1, 9, 8, 6]**

**p4\_table = [2, 4, 3, 1]**

**IP = [2, 6, 3, 1, 4, 8, 5, 7]**

**IP\_inv = [4, 1, 3, 5, 7, 2, 8, 6]**

**expansion = [4, 1, 2, 3, 2, 3, 4, 1]**

**s0 = [[1, 0, 3, 2], [3, 2, 1, 0], [0, 2, 1, 3], [3, 1, 3, 2]]**

**s1 = [[0, 1, 2, 3], [2, 0, 1, 3], [3, 0, 1, 0], [2, 1, 0, 3]]**

**# key generation**

**temp = apply\_table(key, p10\_table)**

**left = temp[:5]**

**right = temp[5:]**

**left = left\_shift(left)**

**right = left\_shift(right)**

**key1 = apply\_table(left + right, p8\_table)**

**left = left\_shift(left)**

**right = left\_shift(right)**

**left = left\_shift(left)**

**right = left\_shift(right)**

**key2 = apply\_table(left + right, p8\_table)**

**while True:**

**mode = input("Выберите режим (шифрование - 1 /расшифрование - 2): ").strip().lower()**

**if mode != '1' and mode != '2':**

**print("Неверный режим. Пожалуйста, выберите 'шифрование - 1' или 'расшифрование - 2'.")**

**continue**

**if mode == '1':**

**print("Ключ 1: ", key1, "\nКлюч 2: ", key2)**

**temp = apply\_table(message, IP)**

**print(temp)**

**temp = function(expansion, s0, s1, key1, temp)**

**print(temp)**

**temp = temp[4:] + temp[:4]**

**print(temp)**

**temp = function(expansion, s0, s1, key2, temp)**

**print(temp)**

**CT = apply\_table(temp, IP\_inv)**

**print("Cipher text is:", CT)**

**elif mode == '2':**

**print("Ключ 1: ", key1, "\nКлюч 2: ", key2)**

**temp = apply\_table(message, IP)**

**print(temp)**

**temp = function(expansion, s0, s1, key2, temp)**

**print(temp)**

**temp = temp[4:] + temp[:4]**

**print(temp)**

**temp = function(expansion, s0, s1, key1, temp)**

**print(temp)**

**PT = apply\_table(temp, IP\_inv)**

**print("Plain text after decrypting is:", PT)**

