**Лабораторная работа № 5**

**ИССЛЕДОВАНИЕ БЛОЧНЫХ ШИФРОВ**

Цель: изучение и приобретение практических навыков разработки и использования приложений для реализации блочных шиф ров (рассчитана на 4 часа аудиторных занятий).

Задачи:

1. Закрепить теоретические знания по алгебраическому описанию, алгоритмам реализации операций зашифрования/расшифрования и оценке криптостойкости блочных шифров.

2. Разработать приложение для реализации указанных преподавателем методов блочного зашифрования/расшифрования.

3. Выполнить анализ криптостойкости блочных шифров.

4. Оценить скорость зашифрования/расшифрования реализованных шифров.

5. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

**Практическое задание**

1. Разработать авторское приложение в соответствии с целью лабораторной работы. При этом можно воспользоваться готовыми библиотеками либо программными кодами, реализующими некоторые блочные алгоритмы.

Приложение должно реализовывать следующие операции:

• разделение входного потока данных на блоки требуемой длины с необходимым дополнением последнего блока;

• выполнение требуемых преобразований ключевой информации;

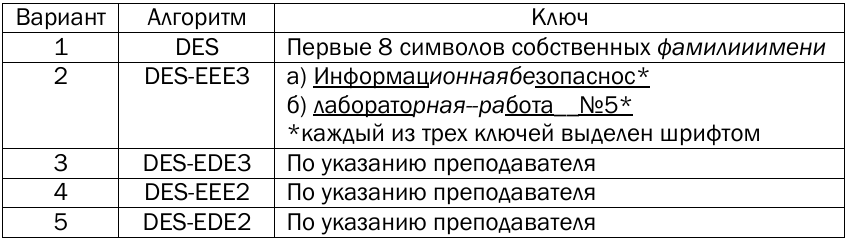
• выполнение операций зашифрования/расшифрования;

• оценка скорости выполнения операций зашифрования/расшифрования;

• пошаговый анализ лавинного эффекта с подсчетом количества изменяющихся символов по отношению к исходному слову.

Исследуемый метод шифрования и ключевая информация – в соответствии с вариантом из табл. 5.1.

Таблица 5.1 – Варианты заданий



Приложение было реализовано на языке python, функции шифрования и расшифрования представлены в листинге 1.1.

def triple\_des\_encrypt(message: str, key1: str, key2: str) -> bytes:

k1 = adjust\_key(key1)

k2 = adjust\_key(key2)

m = pad(message.encode('utf-8'), 8)

cipher = DES.new(k1, DES.MODE\_ECB)

enc1 = cipher.encrypt(m)

cipher = DES.new(k2, DES.MODE\_ECB)

enc2 = cipher.encrypt(enc1)

cipher = DES.new(k1, DES.MODE\_ECB)

enc3 = cipher.encrypt(enc2)

return enc3

def triple\_des\_decrypt(ciphertext: bytes, key1: str, key2: str) -> str:

k1 = adjust\_key(key1)

k2 = adjust\_key(key2)

cipher = DES.new(k1, DES.MODE\_ECB)

dec3 = cipher.decrypt(ciphertext)

cipher = DES.new(k2, DES.MODE\_ECB)

dec2 = cipher.decrypt(dec3)

cipher = DES.new(k1, DES.MODE\_ECB)

dec1 = cipher.decrypt(dec2)

return unpad(dec1, 8).decode('utf-8')

Листинг 1.1 – функции шифрования и дешифрования

Результат выполнения программы представлен на рисунке 1.1.

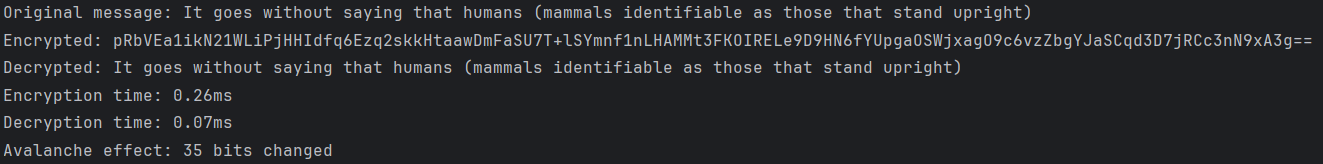


Рисунок 1.1 результат работы программы

2. Проанализировать влияние слабых ключей и полуслабых ключей на конечный результат зашифрования и на лавинный эффект.

Слабые ключи – это такие ключи для DES, для которых все раунды алгоритма производят идентичные (или взаимно инвертирующие) преобразования. В результате при шифровании получаются аномалии. Например, при использовании слабого ключа два последовательных применения DES (шифрование, а затем повторное шифрование) могут оказаться тождественными операциями. Это приводит к тому, что шифрованный текст может быть недостаточно хаотичным, а криптосистема становится уязвимой к криптоанализу.

Полуслабые ключи представляют собой пары ключей, обладающие следующим свойством: если зашифровать сообщение одним из ключей пары, то дешифровать его можно, используя второй ключ той же пары. То есть при использовании полуслабых ключей операция зашифрования одной парой может служить взаимно обратимой операцией с использованием другого ключа из пары. Это снижает стойкость алгоритма, так как криптоаналитик может, обнаружив такую пару, сузить область возможных ключей.

Сложность ключа практически никак не влияет на лавинный эффект, на лавинный эффект влияет свойство алгоритма шифрования.

Функция лавинного эффекта представлена в листинге 1.2.

def avalanche\_effect(message: str, key1: str, key2: str, byte\_to\_flip: int, bit\_to\_flip: int) -> int:

original\_cipher = triple\_des\_encrypt(message, key1, key2)

m\_bytes = bytearray(message.encode('utf-8'))

if byte\_to\_flip >= len(m\_bytes):

raise ValueError("Byte index out of range")

m\_bytes[byte\_to\_flip] ^= (1 << bit\_to\_flip)

modified\_message = m\_bytes.decode('utf-8', errors='replace')

modified\_cipher = triple\_des\_encrypt(modified\_message, key1, key2)

return count\_diff\_bits(original\_cipher, modified\_cipher)

Листинг 1.2 – функция лавинного эффекта

3. Оценить степень сжатия (используя любой доступный архиватор) открытого текста и соответствующего зашифрованного текста. Дать пояснения к полученному результату.

Исходный текст сжимается лучше, чем зашифрованный, это можно объяснить тем, что при шифровании данные преобразуются в форму, которая зачастую неэффективно оптимизируется для хранения. Это противоречит логике сжатия, целью которого является уменьшение объёма данных за счёт их структуры и повторов. Так же при шифровании возрастает энтропия.

Результаты сжатия представлены на рисунке 1.2.

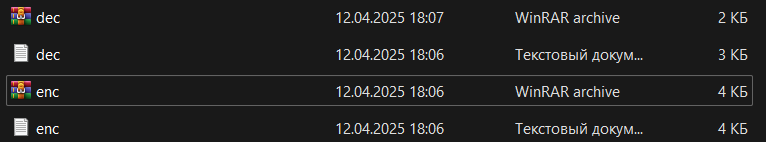


Рисунок 1.2 результат сжатия текстов

4. Результаты оформить в виде отчета по установленным правилам.