Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Информационная безопасность

Студент: Чёрная Я.Р.

ФИТ 3 курс 4 группа

Преподаватель: Нистюк О. А.

Минск 2025

**Лабораторная работа №7. Исследование блочных шифров**

**Цель:** изучение и приобретение практических навыков разработки и использования приложений для реализации блочных шифров.

**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.

**Теоретические сведения**

Все стандарты и алгоритмы блочных шифров (БШ) строятся на основе подстановочных и перестановочных шифров, т. е. являются комбинационными. БШ относятся также к классу симметричных.

Блочное зашифрование (расшифрование) предполагает разбиение исходного открытого (зашифрованного) текста на равные блоки, к которым применяется однотипная процедура зашифрования (расшифрования). Указанная однотипность характеризуется прежде всего тем, что процедура зашифрования (расшифрования) состоит из совокупности повторяющихся наборов преобразований, называемых раундами.

В основе криптостойкости блочных шифров лежит идея К. Шеннона о представлении составного шифра таким образом, чтобы он обладал двумя важными свойствами: рассеиванием и перемешиванием. Рассеивание должно скрыть отношения между зашифрованным текстом и исходным текстом. Рассеивание подразумевает, что каждый символ (или бит) в зашифрованном тексте зависит от одного или всех символов в исходном тексте. Другими словами, если единственный символ в исходном тексте изменен, несколько или все символы в зашифрованном тексте будут также изменены. Идея относительно перемешивания заключается в том, что оно должно скрыть отношения между зашифрованным текстом и ключом

Алгоритм DES строится на основе сети Фейстеля. Входной блок данных, состоящий из 64 битов, преобразуется в выходной блок идентичной длины. В алгоритме широко используются рассеивания (подстановки) и перестановки битов текста. Комбинация двух указанных методов преобразования образует фундаментальный строительный блок DES, называемый раундом или циклом.

Один блок данных подвергается преобразованию (и при зашифровании, и при расшифровании) в течение 16 раундов. После первоначальной перестановки и разделения 64-битного блока данных на правую (R0) и левую (L0) половины длиной по 32 бита выполняются 16 раундов одинаковых действий.

Модификацией DES является 3DES. Он был создан У. Диффи, М. Хеллманом и У. Тачманном в 1978 г.

Формальная запись:

Ci = f (Mj, (DES(K3, (DES(K2, (DES, K1)))))).

Существуют несколько реализаций алгоритма 3DES. Вот некоторые из них:

* DES-EEE3: шифруется 3 раза с 3 разными ключами (операции шифрование-шифрование-шифрование);
* DES-EDE3: 3DES операции шифрование-расшифрование-шифрование с разными ключами;
* DES-EEE2 и DES-EDE2: как и предыдущие, однако на первом и третьем шаге используется одинаковый ключ.

Расшифрование происходит, как и в простом DES, в обратном порядке по отношению к процедуре зашифрования.

3DES с тремя ключами реализован во многих Интернет-приложениях. Например, в PGP (Pretty Good Privacy) он позволяет выполнять операции шифрования и цифровой подписи сообщений, файлов и другой информации, представленной в электронном виде, например, на жестком диске; в S/mime применяется для обеспечения криптографической безопасности электронной почты.

3DES используется при управлении ключами в стандартах ANSI X9.17 (метод генерации 64-битных ключей) и ISO 8732 (управление ключами в банковском деле), а также в PEM (Privacy Enhanced Mail).

**Ход работы**

**Задание 1.** Разработать авторское приложение в соответствии с целью лабораторной работы. 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.

Для выполнения шифрования и дешифрования с помощью алгоритма DES использовалась библиотека node-forge.

Функция encrypt принимает на вход шифруемый текст, ключ и выполняет зашифрование.

exports.encrypt = (msg, key) => {

    const cipher = forge.cipher.createCipher('DES-ECB',

forge.util.createBuffer(key));

    cipher.start();

    cipher.update(forge.util.createBuffer(msg, 'utf8'));

    cipher.finish();

    return cipher.output.toHex();

};

Листинг 7.1 – Функция encrypt

Для дешифрования была разработана функция decrypt.

exports.decrypt = (encryptedHex, key) => {

    const encryptedBytes = forge.util.hexToBytes(encryptedHex);

    const encryptedBuffer = forge.util.createBuffer(encryptedBytes);

    const decipher = forge.cipher.createDecipher('DES-ECB',

forge.util.createBuffer(key));

    decipher.start();

    decipher.update(encryptedBuffer);

    decipher.finish();

    return decipher.output.toString();

};

Листинг 7.2 – Функция decrypt

Функция avalancheEffect принимает исходный текст и шифрует его. Затем изменяет один бит исходного текста, снова шифрует его и подсчитывает количество измененных бит и процентах.

exports.avalancheEffect = (input, key1, key2, key3) => {

    const originalEncrypted = exports.encrypt(input,key1,key2,key3);

    const originalBinary = stringToBinary(input);

    const modifiedBinary = invertLastBit(originalBinary);

    const modifiedString = binaryToString(modifiedBinary);

    const modifiedEncrypted = exports.encrypt(modifiedString, key1,

key2, key3);

    const a = hexToBinary(originalEncrypted);

    const b = hexToBinary(modifiedEncrypted);

    let changes = 0;

    for (let i = 0; i < a.length; i++) {

        if (a[i] !== b[i])

changes++;

    }

    const percent = ((changes / a.length) \* 100).toFixed(2);

    return {

        original: input,

        modified: modifiedString,

        avalancheEffect: percent

    };

};

Листинг 7.3 – Функция avalancheEffect

Результат работы приложения на рисунке 7.1.

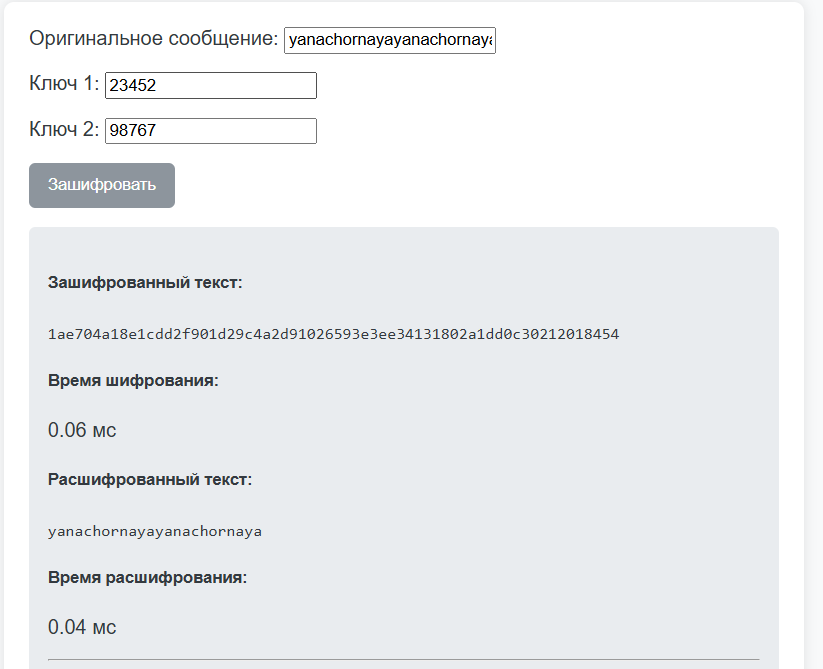


Рисунок 7.1 – Результат шифрования и дешифрования текста

Пошаговый анализ лавинного эффекта в DES-EEE2:

Этап 1. Шифрование (DES-Encrypt с K1):

1. Оригинальное сообщение: yanablack.
2. Применяем алгоритм шифрования с первым ключом.
3. Получаем зашифрованный текст.
4. Сравниваем двоичное представление оригинального зашифрованного текста и изменённого текста.

Этап 2: Шифрование (DES-Encrypt с K2)

1. Применяем алгоритм шифрования со вторым ключом.
2. Получаем зашифрованный текст.
3. Сравниваем двоичное представление изменённого текста.

Этап 3: Шифрование (DES-Encrypt с K1)

1. Используем текст, полученный на предыдущем этапе.
2. Применяем алгоритм шифрования с первым ключом.
3. Получаем повторно зашифрованный текст.
4. Сравниваем двоичное представление.

Результат анализа лавинного эффекта представлен на рисунке 7.2.

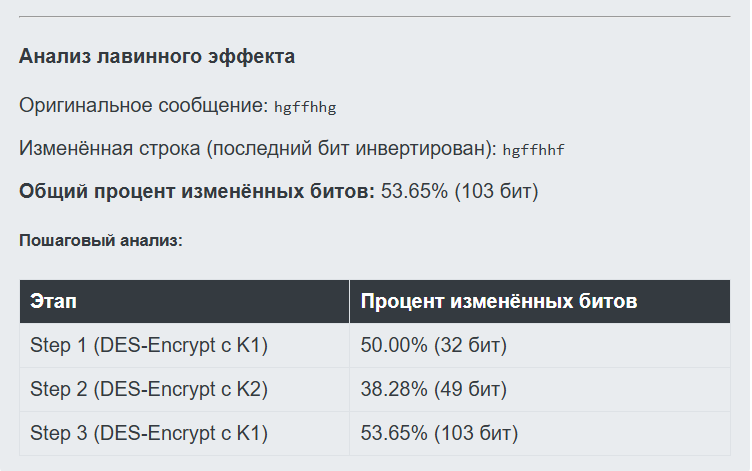


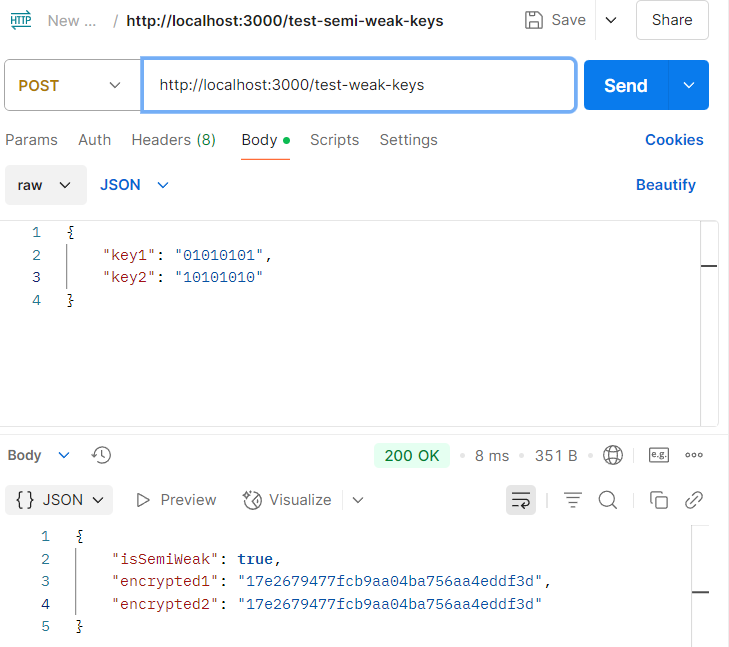
Рисунок 7.2 – Эффект лавины для исходного текста

Лавинный эффект — это явление, при котором небольшое изменение в системе приводит к масштабным последствиям.

При использовании слабых ключей



Можно заметить, что эффект лавины значительно уменьшается, поскольку слабые и полуслабые ключи имеют повторяющиеся или взаимозаменяемые подключи, что приводит к предсказуемым преобразованиям. Из-за этого малые изменения во входных данных слабо влияют на выход, и лавинный эффект значительно ослабевает.



Далее, была проведена оценка степени сжатия открытого и зашифрованного текста. Результат приведен на рисунке 7.5.

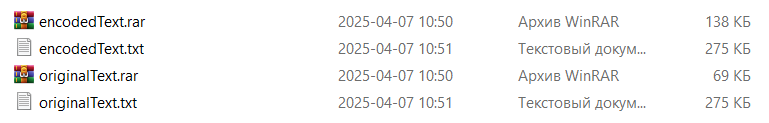


Рисунок 7.5 – Результат шифрования и дешифрования сообщения

Можно заметить, что исходный текст сжался на 75%, в то время как зашифрованный только на 50%. Это происходит потому, что шифрование преобразует информацию таким образом, что создается псевдослучайная последовательность битов, к которым практически невозможно применить существующие алгоритмы сжатия.

**Вывод:** в ходе выполнения лабораторной работы был изучен принцип работы блочного алгоритма DES-EEE2. Также было разработано приложение для реализации зашифрования и расшифрования с помощью этого алгоритма. Также была оценена скорость шифрования и расшифрования данным алгоритмом.